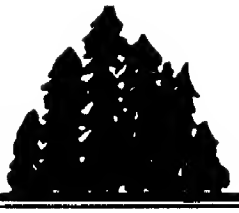
NORTHERN BYTES



Volume 6 Number 2

Greetings! Perhaps you have already heard of the government weather control experiment that was based right here in Sault Ste. Marie this year. The idea was to divert the jet stream farther north, thus keeping most of the snow and extremely cold weather in northern Canada, and permitting a longer shipping season through the Sault locks and on Lake Superior. The entire apparatus was controlled by several Z-80 based microcomputers. The experiment was a dismal failure, and no one could figure out why, until it was discovered that someone had programmed a machine language JR C instruction (jump if Carry flag is set) where there should have been a JR NC instruction (jump if Carry flag is NOT set). The net result of this syntax error was that instead of diverting the cold weather to the north, it pushed all the snow and cold that Sault Ste. Marie usually gets to the south instead. Rumor has it that the programmer in question owned some stock in a ski resort located just north of Sault Ste. Marie, Ontario, Canada.

At one point the computer was instructed to divert a particularly bad storm to the far north, near James Bay. Unfortunately, the programming error caused it to divert the storm to the south instead, so that the worst part of the storm centered on an area near Tullahoma, Tennessee. Anyway, the folks here in Sault Ste. Marie apologize for sending our weather south, but the public relations people in Washington are still insisting that it was a computer foul-up (when they will even condescend to talk about it at all). Even so, government funding for the program has been cut, which means we won't be able to divert any of our cool northern air to the south this summer. Sorry about that.

Well, if this issue of NORTHERN BYTES experiences the usual publication and mailing delays, you should be seeing some spring weather by the time you get this issue. In fact, this issue ould reach you on or about the First of April. Keep that date in nd if you find the information in the above paragraphs a bit hard to swallow!

On to the real world. Winter is the time of year when it's most convenient for us computer nuts to stay indoors and hack away. One by-product of this is that I get MANY more article submissions in the winter than in the warmer times of the year. So, if you've sent an article recently and it hasn't appeared yet, please be patient - it will probably appear in the not too far distant future. On the other hand, certain articles I have received may never appear, simply because they are too difficult to process.

Folks, I think I've spent paragraphs and paragraphs in NORTHERN BYTES trying to educate folks as to how to make life a little easier for your editor. Why should you care? Because I only have so many hours to devote to working on Northern Bytes. If I find I am spending too much time on it, I may very likely call the whole thing off! And even if I continue, believe me when I say that you will have a better newsletter if you can just help me out a little. So, here's this month's DO's and DON'Ts of newsletter article submission!

1) DON'T submit articles or program listings on paper (or other hardcopy). Unless they're VERY short (one or two paragraphs), it takes me a VERY LONG time to re-type them. Please - send a disk, send a tape, send me some MCI Mail, or send your secretary up to do the typing!

2) DON'T forget to put the TEXT FILE on your disk. This includes the text file of the cover letter you send me, if any (particularly since the cover letter is the only way some of you document your programs!). It's very frustrating to get all the program listings on a disk, along with PRINTED documentation. I can use text from virtually any word processing program (please save the text file in "ASCII format" if your word processor gives you that option, especially if you use SuperScripsit). DON'T RGET THE TEXT FILE!

3) VERY IMPORTANT - DON'T send your articles to The Alternate Source - DO send them directly to NORTHERN BYTES in Sault Ste. Marie (the address can always be found on the back page, in the "return address" area). If you ignore this "DON'T", the chances are very good (for various reasons that are not necessarily the fault of TAS) that your articles will never be published at all. So if you've gone to all the trouble to write an article for NORTHERN BYTES, don't (in effect) throw it away by sending it to Lansing - send it directly to NORTHERN BYTES!

4) Here's a DO - now that you've decided to send a disk, PLEASE DO resist the urge to fill it up! If you decide to put a few "odds and ends" on the disk, it means that I have to go through the entire disk to figure out what's what, and that steals time that I could otherwise be using to prepare the newsletter (not to mention the fact that it leaves me with a rather negative impression of the person who sent the disk)! If you do send more than one submission for Northern Bytes, please be sure that EACH submission has a text file and that I have some way of knowing which files go with which articles.

5) DO be sure to SPECIFICALLY STATE if you want your disk returned. I assume you do NOT unless you say you do. And even then, please allow at least a month turnaround time (I am usually lucky if I am ONLY a month behind!). And put a label with your name and address on the disk, so that if it accidentally gets mixed in with other disks, I know which is yours.

6) Once your article has been published, DO check the mailing label on that issue and see if I have credited you with a six-month extension to your "subscription". If I have not, check the label on the NEXT issue you receive. If you still haven't received credit, drop me a line and chew me out!

Most of the above is just "common sense" - but you'd still be surprised how many folks seem to think that I enjoy re-typing material. As I've stated before, this is a one-person operation, I don't have a secretary, and I do NOT enjoy re-typing letters or articles!

If you ignore any of the above, you run the risk of having your article submission ignored - not necessarily intentionally, it just happens that way sometimes. The two VERY WORST things you can do are to leave something important off of your submission disk (or tape), - either the text file or the program listing - or to send your submission to The Alternate Source instead of to NORTHERN BYTES (by the time it reaches me, IF it reaches me, nobody's really sure what your intentions for the disk were - that's especially true if the disk and your cover letter and/or documentation get separated, as sometimes happens).

Speaking of The Alternate Source - some of you may have heard a rumor that TAS went bankrupt. Well, as you can see, TAS is still in business. What in fact happened was that TAS filed for a "Chapter 11" bankruptcy, which is basically a debt relief measure. TAS customers did not feel any impact at all from this - it was "business as usual" from the customer's standpoint. What it does mean is that some of the folks to whom TAS owed money will have to wait a little longer, and can't come around and start hauling away the fixtures. Hopefully, everyone to whom TAS owes anything will get paid the full amount owed them in the not too far distant future. Once again, I want to emphasize that no TAS customer lost anything, and that TAS is still very much in business. That's a good thing for NORTHERN BYTES readers, too, because even though TAS doesn't own NORTHERN BYTES, if they were ever to discontinue operations, I can assure you that I do NOT have the resources to publish this newsletter myself.

It seems that many of the firms that have offered support for the TRS-80 are in some sort of difficulty these days. I suppose that some of the blame can be laid at Tandy's doorstep (for their lack of support for third-party vendors), but one of the largest expenses for any company is advertising. In the TRS-80 market, there is only one large TRS-80 related magazine, and their advertising rates might be considered prohibitive, especially for the small to medium-size software house. NORTHERN BYTES at present has a circulation of approximately 1500 copies per issue (although that's NOT audited, you'll just have to take my word for it) and we exchange publications with most of the TRS-80 user groups (the ones we know about, anyway). Right now we charge $60.00 for a full page ad (and will sell space in 1/6 page

increments). That rate will increase if our circulation increases, but for right now, where else can you buy 1/6 of a page in a publication directed specifically to TRS-80 Model I/III/4/4P users for only $10? Perhaps the fact that we do sell ad space may be one of our best-kept secrets, but we really do!

That's enough of my rambling for this month. I'll close by once again reminding you that if you want to send us your VISA or Mastercard number and expiration date, we will add you to the NORTHERN BYTES mailing list and automatically send you (and bill you $2.00 for) each new issue. If you live outside North America, please state whether you want airmail delivery (which costs $1.00 extra). If we get enough of these, who knows, we might be able to publish more regularly!

THE EXTERMINATOR

Most of the BUGS reported this month are in the "letters to the editor", but I must mention that I jumped the gun in this column back in Volume 5 Number 8, when I reported that M.D. Kerby had problems with Don Brate's program that patches TRSDOS 1.3. Well, it turns out that M.D. got bitten by the infamous TYPO bug (that's the one that strikes when you've spent too many hours typing at a stretch. It strikes all computer users every now and then, and unfortunately, there doesn't seem to be any immunity). Once he got a copy of TRSPATCH/BAS off of TAS Public Domain Library disk # 004, he had no further problems, and reports that TRSPATCH/BAS does everything claimed for it. So, please ignore the warning about TRSPATCH/BAS in Volume 5, Number 8!

LETTERS DEPARTMENT

Neither rain, nor snow, nor sleet, nor gloom of night shall prevent these letters from arriving here at NORTHERN BYTES (but when we get four feet of snow on the ground, it tends to slow them down a little!):

Date: Mon Jan 14, 1985 1:17 am EST **RECEIPT
From: Anthony J. Domigan / MCI ID: 254-5121

TO: *Jack Decker / MCI ID: 102-7413
Subject: T.Domigan - Letter

Jack,

Here are a couple of tidbits that may be useful to other Northern Bytes readers.

In NEWDOS/80 version 2 for the Model 3 there is a handy SYS0 permanent routine to convert a byte pair or a single byte to its ASCII equivalent via a DAA routine.

With the byte pair to display in DE and the buffer address in HL, a call to 44D2H will load the buffer with the ASCII of the DE byte pair.

```
e.g.  LD    HL,3C00H    ;BUFFER - IN THIS CASE DIRECT TO SCREEN
      LD    DE,1234H    ;BYTE PAIR TO DISPLAY
      CALL  44D2H       ;DISPLAY 2 HEX BYTES 12 AND 34 TO SCREEN
```

A single byte may also be converted by entering this routine a little further on, at 44D7H, with the value to convert in A and the buffer pointed to by HL again.

```
e.g.  LD    HL,3C00H    ;BUFFER - IN THIS CASE DIRECT TO SCREEN
      LD    A,2FH       ;HEX BYTE TO DISPLAY
      CALL  44D7H       ;DISPLAY HEX BYTE 2F TO SCREEN
```

These routines are handy as they can be used without concern for DOS links and are permanently available like the ROM.

For Video4(KB) users here are some trivial enhancements to Newdos. I modified the 'NEWDOS/80 READY', the 'DISK BASIC' and the 'READY' prompts. Did I hear a groan? Yes, I know its been done many times before, but this is a little different (I defend). I modified them all to display in reverse video and for the prompts to give a 'bell'.

The 'NEWDOS/80 READY' can be found in SYS1/SYS,04,E0.

```
find 4E45 5744 4F32 2F38 3020 5245 4144 590D
chg  104E 4557 444F 5320 5245 4144 5911 070D
```

The DISK BASIC LOGO can be found in BASIC/CMD,16,79. The code begins with 1C 1F DISK BASIC 20 20 etc. and I changed it to 1F 10 DISK BASIC 11 0D. This saves an unnecessary CLS, and substitutes a neat and succinct Reverse Video DISK BASIC message.

The last patch is to ROM BASIC's 'READY' prompt. Model 4P users, like myself, can change the MODELA/III file or both Model 4 and 4P users could temporarily make the change with this short routine.

```
        ORG     5200H
START   DI
        LD      A,05H       ;Open ROM (in 80x24)
        OUT     (84H),A     ;
        LD      BC,05H      ;5 chars to move
        LD      HL,CHG      ;Point to new bytes
        LD      DE,1929H    ;ROM patch point
        LDIR
        LD      A,04H       ;80x24
        OUT     (84H),A     ;Reinstate ROM
        EI
        JP      402DH
CHG     DEFB    10H         ;Reverse video
        DEFM    'OK'        ;New prompt
        DEFW    0711H       ;Bell & normal video
        END     START
```

Fortunately, these changes do not cause any ill effects in Newdos when Video4(KB) is not active, so they are reasonably harmless mods to the DOS.

Tony Domigan
P.O. Box 150, Thomastown, Victoria, 3074, Australia

[Thanks for passing along this info, Tony. I did some checking and it turns out that Model I users can also make use of the NEWDOS/80 hexadecimal display routines you mentioned, but they are at different locations in the Model I version of NEWDOS/80. The equivalent starting addresses in the Model I version are 4063H and 4068H (instead of 44D2H and 44D7H respectively).]

Dear Jack,

Hi. I'm the Editor of the INTERFACE, the Newsletter of the San Gabriel Valley TRS-80 Users Group (SAGATUG). Our Preside Andy Levinson, came upon Volume 5, #1-7, of NORTHERN BYT. (although I'm not sure where he got them). He knows how desperate I am for material to put in the INTERFACE, so he gave them to me.

I am very impressed with NORTHERN BYTES. There is a wealth of useful information in there. I plan on reprinting several of the articles... with proper credit of course. I'm glad to have some quality material to publish, and delighted that you have made it available for reprinting.

Our club is a TRS-80 only organization. We seem to be unusual in that our club is thriving, in a time when TRS-80 related clubs (and magazines) seem to be disappearing. Our membership is about 100. Most members have Model III's and 4's - the 4P is extremely popular. A few members are still using Model I's. I bought my Model I in 1978, and I still use it to access the BBS systems.

I was particularly interested in one article in Volume 5 Number 7 - the directions for making a NEWDOS/80 disk that loads the ROM image and auto-boots on the 4P. In fact, I think most 4P owners who use NEWDOS/80 would be very interested, especially since Apparat seems totally disinterested in supporting the Model 4 (at least that's what they told me when I called them to ask!!). We had just discussed making an autobooting NEWDOS/80 disk at the last club meeting, and concluded it would not be possible because of the hardware in the 4P. Obviously we were wrong!!

I followed the directions as outlined, but they wouldn't work at first. I think there are a few typos in the article.

First, obviously, the NDP=NEWDOS4P in the copy instruction should be NDN (new diskette name). Then, the bytes to change in the MODELA/III file are at Sector 53, relative byte A9, not relative byte 39 as stated in the article.

In addition, if you zap a length of 9 grans for the ROM image, then you can't change sector 53 - SUPERZAP quits at relative sector 50 with a "Can't extend file via read" error. The MODELA/III f is 57 sectors long, which is almost 12 grans (5 sectors/gran). A trusty copy of "TRS-80 Disk and Other Mysteries" tells me that byte is the # grans - 1, so it should be 0C - 1 = 0B. I'm not sure why he shortens the file to 09. Anyway, if you leave it 0B, then SUPERZAP allows the changes at sector 53.

With those changes, it WORKS!! The NEWDOS/80 disk loads the ROM and boots. BUT (there's always a BUT), if you make a

backup using COPY :0 :1 00/00/00, the copy won't boot. It seems that NEWDOS/80 re-writes track 1, sector 0, rather than copying the nice zero'd and zapped sector we made. You have to use SUPERZAP on the copy's track 1, sector 0. Then it works OK.

Also, if you DO zap the length to 09 as mentioned in the article, it also boots OK. But if you try to backup the disk, the copy quits with an "ERROR IN SECTOR 50, FILE MODELA/III" message. So I think it's better to leave the length 0B.

Now for MY problem. My 4P has a 40/40 double headed drive as drive 0, so I want to make an auto-booting two-sided disk. I've been trying, but no luck yet. I'll describe what I've done so far, and maybe you can help.

Double sided drives have 36 sectors per true track, so I made a disk with the directory on sector 720 (DDSL = 72). SUPERZAP verified that the directory really ended up on true track 20. So far so good. Next, I CREATEd MODELA/III on that disk and zapped its FPDE. I wanted the file to copy over to true track 30. Since there are 36 sectors per true track, and 10 sectors per relative track, true track 30 * 3.6 = relative track 108. This is 6C hex. So I zapped the FPDE with 6C0B, and copied over MODELA/III. I checked with SUPERZAP - the file started on true track 30, sector 0. That seemed to be OK. I changed the GAT entries at 6C and following to FF to show the file was there. I zapped the MODELA/III file at A9 to get NEWDOS to boot from track 0, and zero'd and modified track 1, sector 0 as in the article. Finally I zapped the FPDE again (as given in the article) to show the file starting at 1E (apparently the hardware reads the directory in true tracks when loading the ROM image, but NEWDOS/80 COPY reads the directory in relative tracks, and 1E0B shows the file starting at track 30).

So it SHOULD work. When I boot it, though, the ROM begins to load (I get the "ROM loading" message, and I hear the drive stepping), but, just at the end of the load, just at the point I expect it to boot NEWDOS, I get the message "Can't load ROM Image - Too Many Extents" in three languages!!! I can't figure it out - I only have the one extent in the directory. The rest are FF, just as they're supposed to be. Any ideas?

Well, I'll let you get back to your hacking. As Editor of a newsletter myself, I know how much effort goes into what you're doing. I just want you to know that those of us who are using, enjoying, (and depending on) our TRS-80s really appreciate the kind support NORTHERN BYTES provides. I'm not smart enough to patch the ROM myself, but I'm glad there are people who are, and I'll be happy to spread the information as best I can.

John T. Phillipp
815 South Walnut, San Dimas, California 91773

[John, I'm glad to hear that you've found NORTHERN BYTES a valuable source of information for your user group newsletter! I passed your letter along to Tony Domigan (author of the article in question), and his reply follows:]

Jack,

Following John Phillipp's enquiry I re-examined my 4P auto-ROM boot patch for NEWDOS/80. I worked through a couple of part-patches before the most logical course became obvious to me.

The solution was to prepare a NEWDOS/80 system disk with a directory pointer satisfactory to both Newdos and the on-board loader (which will be looking for an LDOS DIR pointing to the MODELA/III file).

I configured NEWDOS with a directory on lump 29 or sector 290. The on-board loader will read the DIR pointer from the boot sector and interpret this as true track 29 or sector 522.

A MODELA/III FPDE is created and patched to load to track 30 or sector 360. MODELA/III is copied from TRSDOS to the new disk and is loaded to track 30 onwards.

The NEWDOS directory is copied from lump 29 to track 29 for the on-board loader to read. The moved FPDE for MODELA/III is modified to reflect the track and gran count pertinent to an LDOS FPDE of this file.

The original FPDE in lump 29 of NEWDOS is modified to show the MODELA/III file starting at track 29, adding the extra length to the gran and sector count so that as far as NEWDOS is concerned is a contiguous file block. This file contains The LDOS-type MODELA FPDE followed by some spare space (care to store these instructions there?) and then followed by the original unmodified MODELA/III.

It's nearly all done except to fix-up the GAT. Running DIRCHECK tells us which bytes of the GAT must be modified to allocate the extra grans the new MODELA/III length.

With that it is done. This configuration is acceptable to both NEWDOS and the on-board loader and importantly the usual NEWDOS backup, i.e. COPY,0,1,,FMT etc. will properly back-up the disk.

REVISED PATCH TO ALLOW NEWDOS/80/v2 TO AUTO-BOOT MODELA/III.

Assuming system disk in drive 0 and destination disk in drive 1.

1. PDRIVE,0,1,DDSL=29,A<CR>

2. COPY,0,1,02/02/85,CBF,/SYS,FMT<CR>

Assuming new system disk on drive 0.

3. CREATE MODELA/III:0<CR>

4. Using SUPERZAP modify MODELA/III FPDE - Rel.Sector 299 for me

```
find
1000 0000 004D 4F44 454C 4120 2049 4949 .....MODELA..III
9642 9642 0000 FFFF FFFF FFFF FFFF FFFF .B.B............

change to
1000 0000 004D 4F44 454C 4120 2049 4949 .....MODELA..III
9642 9642 0000 3600 FFFF FFFF FFFF FFFF .B.B..6.........
```

5. COPY MODELA/III:1 :0 SPDN=4<CR> ' if 4=TRSDOS for you

6. Using SUPERZAP CDS Copy 10 Disk Sectors from 290 to 522

7. Using SUPERZAP modify the transferred FPDE for MODELA/III

```
from (sector 531 on my disk)
1020 0000 004D 4F44 454C 4120 2049 4949 .....MODELA..III
9642 9642 3900 360B FFFF FFFF FFFF FFFF .B.B9.6.........

to
1000 0000 004D 4F44 454C 4120 2049 4949 .....MODELA..III
9642 9642 3900 1E09 FFFF FFFF FFFF FFFF .B.B9...........
```

8. Using SUPERZAP modify the REAL FPDE of MODELA/III

```
from (sector 529 on my disk)
1020 0000 004D 4F44 454C 4120 2049 4949 .....MODELA..III
9642 9642 3900 360B FFFF FFFF FFFF FFFF .B.B9.6.........

to
1020 0000 004D 4F44 454C 4120 2049 4949 .....MODELA..III
9642 9642 4D00 340F FFFF FFFF FFFF FFFF .B.BM.4.........
```

9. Execute DIRCHECK to check for the lumps to be allocated (following the step 8 lengthening of the Newdos file). My readout was:

```
NEWDOS4P 00/00/00
34,0 xxxxx GRANULE FREE, BUT ASSIGNED TO FILE(S) 67 MODELA/III
34,1 xxxxx GRANULE FREE, BUT ASSIGNED TO FILE(S) 67 MODELA/III
35,0 xxxxx GRANULE FREE, BUT ASSIGNED TO FILE(S) 67 MODELA/III
35,1 xxxxx GRANULE FREE, BUT ASSIGNED TO FILE(S) 67 MODELA/III
36,0 xxxxx GRANULE FREE, BUT ASSIGNED TO FILE(S) 67 MODELA/III
```

10. Using SUPERZAP go to the GAT sector (290) and modify relative bytes indicated by DIRCHECK.

e.g. relative byte 34 = FC change to FF i.e. allocate both grans.
relative byte 35 = FC change to FF i.e. allocate both grans.
relative byte 36 = FE change to FF i.e. allocate 1st gran also.

11. Reboot NEWDOS holding down the 'L' key - it should load the MODELA/III file and execute NEWDOS/80 successfully.

Tony Domigan
P.O. Box 150, Thomastown, Victoria, 3074, Australia.
* MCI-ID : 254-5121 * SOURCE-ID : BCT039 *

Dear Jack Decker,

... [In order] to optimize the contact between TRS-80 users, I'd like to found a mailbox for TRS-80 users on packet-switching network. This will make it possible for TRS-80 users to contact other TRS-80 users world wide at an acceptable charge. What do you think about this? Perhaps you can note this inside your newsletter.

You may send me a letter into my mailbox, please use (Datex-p, West Germany) NUA 45 6673 30070,ZCZCGB address your letter to G,SENDER, password: SENDER.

I'm looking forward to your answer.

Gerd Sender
Moselstrasse 39, 5000 Koeln 50, West Germany

[I like your idea about having a contact for TRS-80 users on a packet-switching network. I've thought about doing something like this myself, but so far I have not found any cost-effective way to do it.

As for your "Datex-p" mail address, unfortunately I do not have enough information to be able to access it. As far as I can determine, in order to be able to access it, I would have to have a prepaid account on one of the U.S. packet switching networks. Very few personal computer users have such a thing. Here in the U.S., the host computer normally pays the network charges, then bills them back to the user. For example, Compuserve Information Service permits access via Telenet or Tymnet, but bills $2.00 per hour extra for access via these networks. Thus the user does NOT have to have a prepaid account with Telenet or Tymnet. Furthermore, as I understand it, some countries (such as Canada) simply do not offer user-prepaid accounts on their data networks (in Canada, Bell Datapac will bill the host computer only at present).

NORTHERN BYTES is in a particularly strange situation, due to being at some distance from any major U.S. city. Thus, we are totally unable to access Tymnet, without paying a long-distance telephone charge. Telenet and Autonet are available to us over toll-free ("800" area code) numbers, but these usually carry an additional surcharge (which is usually higher than the rate for a regular long-distance telephone call to the host computer!). Thus, I am not a subscriber to any of the packet networks or services such as Compuserve, The Source, etc. I am hooked up to MCI Mail, an electronic mail service here in the U.S. that is also available worldwide. Unfortunately, MCI Mail does not support "bulletin board" type operation - mail can only be sent from one user to another - and even it can get rather expensive, at $1.00 per "computer letter" (plus, I have to pay 15 cents per minute to access MCI's 800 "toll-free" number). My MCI Mail ID is 102-7413. I also have a TELEX number through this service, which is 6501027413 (answerback is 6501027413 MCI). Unfortunately, that is the only electronic mail service that I can access at present!

But I do like the concept, and if you can find a cost-effective way to do it, please let us know!]

Dear Jack,

... In conjunction with [a] project which is to be done on a Model II, I ran into some interesting facts. Apparently Logical Systems worked up a Model II/12 version of TRSDOS 6 but then never marketed the product. My guess (sheer speculation, mind you) is that they tried to sell the idea to Tandy and they wouldn't buy. Anyway, it is possible to get a copy of this DOS, and here's how you do it: First, call or write Logical Systems and tell them you're interested in the DOS. They will send you a license agreement to sign and return with $100. For this you get LS-DOS 6.2, runable on a Model II or 12 and functionally 97% identical to the Model 4 version. You also get about a half-dozen pages of documentation explaining, I guess, the other 3 per cent, and this plus the Model 4 Technical Reference Manual should be all you need to know. It doesn't come with a BASIC, but you can get a copy of Model 4 BASIC on it and it will run fine. Anyway, the catch to this whole thing is that, since you're getting the fruits of an abandoned project ... well, remember that license agreement I mentioned? You guessed it: you get absolutely NO support from LSI! The only support is via other users on the LSI SIG of CompuServe. This strikes me as a new high in audacity, but the funny part is, the idea of being able to write software on my Model 4 for the Model II or 12 is so attractive I may well buy the product in spite of this, if a quick visit to CompuServe turns up some people who are successfully working with it. And I'm sure that's just what LSI is banking a lot of other people will do. I can now see why LSI and Tandy make such good bedfellows: they both know when they gotcha!

Bob Grommes
1733 Eastern S.E., Grand Rapids, Michigan 49507

[Thanks, Bob, for passing along this information. If any NORTHERN BYTES readers have had any experience with this version of LS-DOS, I'm sure Bob would appreciate hearing from you!]

Dear Jack!

In Vol. 6, No. 1, B. Jim Smith requests help on understanding TRSDOS 6.XX commands like Set, Route, Filter and Link. You are not given to publishing book reviews but, perhaps, if I make it succinct, you'll make an exception.

Book Review of "The Programmer's Guide to LDOS/TRSDOS Version 6" written by Roy Soltoff, published by MISOSYS, INC., P.O. Box 239, Sterling, Virginia 22170-0239. (205 Pages). This is the BEST book I've ever seen on microcomputers. It is written in (REAL, plain vanilla, easy to understand) English, it is full of examples, it discusses EVERY aspect of the DOS and how to interface to it. It costs $14.95 plus $2 postage. If you want to do ANY work with TRSDOS 6, you should BUY THIS BOOK. End of Book Review. *

Sincerely, Nate Salsbury
610 Madam Moore's Lane, New Bern, North Carolina 28560

[Thanks for the information, Nate. We DO occasionally publish book reviews, but I don't encourage them because they tend to take up space that could be filled by articles that would appeal to a larger percentage of our readers. Your review was just the right length!]

Dear Jack,

Funny people, you Americans. You drive on the wrong side of the road, you talk funny, you have Christmas in winter when it is summer, but worst of all, you write your dates backwards. This is confusing.

We do it proper - DD/MM/YY. Mixing both is confusing, specially in the first twelve days of a month. There is no way to escape as we communicate outside of our computers.

I spent quite some time making my NEWDOS/80 work in th. civilized manner, no mean trick. Now SETDATE has put me back many years. Nothing needs to change in the programme but putting the date to memory, switching the MM and the DD. Not a machine language expert, I cannot find where it's at. If it is simple, could you give me a clue - or the answer?

Clifford S. Richards
3 Boronia Road, Bellevue Hill, Sydney, N.S.W. 2023, Australia

[Funny you should ask for such a thing, because on TAS Public Domain Library Disk #006 I put a modified version of SETDATE called DATESET, which displays the date as, for example, 30 January 1985 instead of January 30, 1985. But it still uses the original order for month and day bytes in reserved RAM. You have apparently patched your copy of NEWDOS/80 to reverse the order of the bytes in which the month and day are stored. So, here are the changes to SETDATE/ASM version 1.3 that should give you what you want. Please let me know how it works!

```
Old:
6028 01      00820      DEFB    1D          ;Day storage
6029 0B      00830      DEFB    11D         ;Month storage

New:
6028 0B      00820      DEFB    11D         ;Month storage
6029 01      00830      DEFB    1D          ;Day storage


Old:
61E1 13      01650      INC     DE          ;Point to day in memory
61E2 1A      01660      LD      A,(DE)      ;Get day from memory
61E3 3D      01670      DEC     A           ;Adjust valid to 0 - 30
61E4 FE1F    01680      CP      31D         ;Is day 1 - 31 in memory?
61E6 3008    01690      JR      NC,USEPGM   ;Go if invalid month
61E8 13      01700      INC     DE          ;Point to month in memory
61E9 1A      01710      LD      A,(DE)      ;Get month from memory
61EA 3D      01720      DEC     A           ;Adjust valid to 0 - 11
61EB FE0C    01730      CP      12D         ;Is month 1 - 12 in mem?
```

```
New:
61E1 13      01650        INC     DE          ;Point to month in memory
61E2 1A      01660        LD      A,(DE)      ;Get month from memory
61E3 3D      01670        DEC     A           ;Adjust valid to 0 - 11
61E4 FE0C    01680        CP      12D         ;Is month 1 - 12 in mem?
61E6 3008    01690        JR      NC,USEPGM   ;Go if invalid month
61E8 13      01700        INC     DE          ;Point to day in memory
61E9 1A      01710        LD      A,(DE)      ;Get day from memory
61EA 3D      01720        DEC     A           ;Adjust valid to 0 - 30
61EB FE1F    01730        CP      31D         ;Is day 1 - 31 in memory?


Old:
6200 0B      01810        DEC     BC          ;Point to month byte
6201 0A      01820        LD      A,(BC)      ;Get month (1 - 12)
6202 C606    01830        ADD     A,6         ;Offset for string table
6204 CDF162  01840        CALL    PRTSTR      ;Print month string
6207 CD1563  01850        CALL    PRTSPC      ;Print space character
620A 0B      01860        DEC     BC          ;Point to day byte
620B 0A      01870        LD      A,(BC)      ;Get day (1 - 31)
620C 6F      01880        LD      L,A         ;Put day in L
620D 2600    01890        LD      H,0         ;HL = day
620F C5      01900        PUSH    BC          ;Save date storage pntr
6210 CD0763  01910        CALL    PRTNUM      ;Print day
6213 C1      01920        POP     BC          ;Restore date storage ptr
6214 CD1063  01930        CALL    PRTCOM      ;Print comma and space


New:
6200 0B      01810        DEC     BC          ;Point to day byte
6201 0A      01820        LD      A,(BC)      ;Get day (1 - 31)
6202 6F      01830        LD      L,A         ;Put day in L
6203 2600    01840        LD      H,0         ;HL = day
6205 C5      01850        PUSH    BC          ;Save date storage pntr
6206 CD0763  01860        CALL    PRTNUM      ;Print day
6209 C1      01870        POP     BC          ;Restore date storage ptr
620A CD1563  01880        CALL    PRTSPC      ;Print space character
620D 0B      01890        DEC     BC          ;Point to month byte
620E 0A      01900        LD      A,(BC)      ;Get month (1 - 12)
620F C606    01910        ADD     A,6         ;Offset for string table
6211 CDF162  01920        CALL    PRTSTR      ;Print month string
6214 CD1563  01930        CALL    PRTSPC      ;Print space character


Old:
62A9 2B      02690 SETDAY DEC     HL          ;Point to month storage
62AA 7E      02700        LD      A,(HL)      ;Get month
62AB CD2663  02710        CALL    MAXDAY      ;Get number days in month
62AE 2B      02720        DEC     HL          ;Point to day storage
62AF 7E      02730        LD      A,(HL)      ;Get current day
62B0 B9      02740        CP      C           ;Compare with maximum
62B1 34      02750        INC     (HL)        ;Advance day of month
62B2 D8      02760        RET     C           ;Finished if valid day
62B3 3601    02770        LD      (HL),1      ;Else first of new month
62B5 23      02780        INC     HL          ;Point to month storage
62B6 7E      02790        LD      A,(HL)      ;Get current month
62B7 FE0C    02800        CP      12D         ;See if it's December
62B9 34      02810        INC     (HL)        ;Advance month count
62BA D8      02820        RET     C           ;Finished if valid month
62BB 3601    02830        LD      (HL),1      ;Else January of new year
62BD 2B      02840        DEC     HL          ;Bump pointer back down
62BE 2B      02850        DEC     HL          ;  to year storage


New:
62A9 2B      02690 SETDAY DEC     HL          ;Bump pointer down to
62AA 2B      02700        DEC     HL          ;  month storage
62AB 7E      02710        LD      A,(HL)      ;Get month
62AC CD2663  02720        CALL    MAXDAY      ;Get number days in month
62AF 23      02730        INC     HL          ;Point to day storage
62B0 7E      02740        LD      A,(HL)      ;Get current day
62B1 B9      02750        CP      C           ;Compare with maximum
62B2 34      02760        INC     (HL)        ;Advance day of month
62B3 D8      02770        RET     C           ;Finished if valid day
62B4 3601    02780        LD      (HL),1      ;Else first of new month
62B6 2B      02790        DEC     HL          ;Point to month storage
62B7 7E      02800        LD      A,(HL)      ;Get current month
62B8 FE0C    02810        CP      12D         ;See if it's December
62BA 34      02820        INC     (HL)        ;Advance month count
62BB D8      02830        RET     C           ;Finished if valid month
62BC 3601    02840        LD      (HL),1      ;Else January of new year
62BE 2B      02850        DEC     HL          ;Point to year storage
```

```
Old:
62D7 2B      02990 SETDA2 DEC     HL          ;Point to month storage
62D8 7E      03000        LD      A,(HL)      ;Get month
62D9 2B      03010        DEC     HL          ;Point to day storage
62DA 35      03020        DEC     (HL)        ;Decrement day of month
62DB C0      03030        RET     NZ          ;Finished if valid day
62DC 3D      03040        DEC     A           ;Else A=# of previous mth
62DD CD2663  03050        CALL    MAXDAY      ;Get # days previous mnth
62E0 71      03060        LD      (HL),C      ;Day=last day prev. month
62E1 23      03070        INC     HL          ;Point to month storage
62E2 35      03080        DEC     (HL)        ;Decrement month count
62E3 C0      03090        RET     NZ          ;Finished if valid month
62E4 360C    03100        LD      (HL),12D    ;Else December prev. year
62E6 2B      03110        DEC     HL          ;Bump pointer back down
62E7 2B      03120        DEC     HL          ;  to year storage


New:
62D7 2B      02990 SETDA2 DEC     HL          ;Bump pointer down to
62D8 2B      03000        DEC     HL          ;  month storage
62D9 7E      03010        LD      A,(HL)      ;Get month
62DA 23      03020        INC     HL          ;Point to day storage
62DB 35      03030        DEC     (HL)        ;Decrement day of month
62DC C0      03040        RET     NZ          ;Finished if valid day
62DD 3D      03050        DEC     A           ;Else A=# of previous mth
62DE CD2663  03060        CALL    MAXDAY      ;Get # days previous mnth
62E1 71      03070        LD      (HL),C      ;Day=last day prev. month
62E2 2B      030B0        DEC     HL          ;Point to month storage
62E3 35      03090        DEC     (HL)        ;Decrement month count
62E4 C0      03100        RET     NZ          ;Finished if valid month
62E5 360C    03110        LD      (HL),12D    ;Else December prev. year
62E7 2B      03120        DEC     HL          ;Point to year storage
```

Those are the changes in the code. You can either change the source code or patch the /CMD file to achieve the desired result. I hope this is what you wanted!]

PATCH FOR TMDD/CMD
Patch supplied by Tony Domigan

Some copies of TMDD (The Memory Disk Driver for the Model 4/4P, sold by TAS) have a minor bug in that when you display a directory of the MEMDISK, the disk name and disk date fields are reversed. This does not really affect operation of the MEMDISK, but if it bothers you, patch TMDD/CMD using SUPERZAP as follows:

TMDD/CMD,02,65 Find: 216D52 Change to: 217552

MODEL 4 ROM & MODEL 4P ROM IMAGE CHANGES
by Jack Decker

This article is a not-too-well organized collection of data on the differences between the "old" and "new" versions of the Model 4 ROM/4P ROM image (that is used when the computer is run in the "Model III mode"). The "new" version was first introduced as the Model 4P ROM image (the "MODELA/III" file supplied with the 4P), then copied back to the "un-portable" Model 4 at about the same time that Radio Shack relocated the arrow keys and started putting green CRT's in the 4 and 4P. The "new" Model 4 ROM and the "new" 4P ROM image have a few minor differences, mostly due to hardware.

The major changes are as follows:

1) Revisions of the ROM in locations 0000H-2FFFH. Although Radio Shack has released several versions of the portion of ROM from 3000H-37FFH, this is the first revision of code in the portion of memory below 3000H since the very early days of the Model III.

2) A revised printer driver with some added character translation capabilities, but this will cause no end of grief for unwary programmers, particularly those that have programs that output graphics characters to the printer.

3) A completely new routine that allows a 4P to be booted off the RS-232 interface.

I'll cover each section individually, but first the credits: Thanks to Nate Salsbury and especially to John Hallgren of Clearwater, Florida for providing the information below. Most of what you see here was extracted from John's handwritten notes on the new ROM, of which he very graciously sent me a copy. Additions, corrections, and clarifications to this article are most welcome. That said, we begin...

A. ROM CHANGES AND ADDITIONS

0001H-0004H Changed in the 4P ROM image only - not on the standard Model 4. Old: XOR A followed by JP 3015H (jump to reset/power-up routine). New: LD A,1 followed by OUT (9CH),A (on the 4P port 9CH is the Boot ROM switch - this code disables RAM/enables ROM boot).

0006H-0007H Changed in the 4P ROM image only - not on the standard Model 4. Old: Part of unused JP 4000H instruction. New: Part of JP 3015H instruction (moved from 0002H, jumps to reset/power-up routine).

0043H-0045H Old: Unused RET instruction followed by two zero bytes. New: JP 0434H instruction, accessed from JP NC instruction at 3790H (part of line printer driver - see commented disassembly).

0063H-0065H Old: Leftover code from the Model I time delay routine (JR NZ,0060H followed by RET). New: JP 041FH instruction, accessed from JP C instruction at 3795H (part of line printer driver - see commented disassembly).

006CH-0074H Old: LD DE,421DH followed by JR 001BH (Start of I/O re-router routine which has been deleted in the Model 4) followed by NOP, and finally followed by an unused JP 06CCH instruction (left over from the Model I ROM code). New: RET instruction (disables I/O re-router) followed by LD BC,1A18H and JP 19AEH instructions and two unused zero bytes. The LD BC,1A18H and JP 19AEH instructions comprise the old Model I "warm re-entry to BASIC" code that was originally found at 06CCH on the Model I, and was erroneously deleted from the Model III. It is used by the BASIC "SYSTEM" command when the BREAK key is depressed (accessed from 02C3H).

00FFH-0101H Old: JP 37EBH (part of routine that prints "Radio Shack Model III Basic" and copyright message. In the Model III, the first instruction at 37EBH was CALL 021BH). New: CALL 021BH (done here instead of at 37EBH).

0103H-0104H Old: Unused JP 1A19H (left over from the Model I). New: JP 01E6H (on the Model 4) or JP 01E7H (on the Model 4P) (jump to remainder of routine to print copyright message).

01D9H-01F7H In the Model I, this block of memory contained the code to output one bit to the cassette. In the "old" ROM, this area contained the screen print routine (plus leftover garbage from the Mod I routine at 01F5H-01F7H). In the "new" ROM, this area contains a vector to the new screen print routine (a JP 3027H which in turn jumps to 37A5H), the printer status test that was formerly located at 0440H-044AH in the "old" ROM (now moved to 01DCH-01E6H in the "new" ROM), code to display the Tandy copyright message and put the BASIC TIME$ vector into place at 4176H (this code was formerly located at 37EEH-37FCH and has been moved to 01E7H-01F5H in the "new" ROM), and unused zero bytes at 01F6H-01F7H. Note that a POP AF instruction is found at 01E5H in the ROM image, but this instruction is deleted in the "new" Mod 4 R and the instructions between there and 01E9H have been moved forward one byte. 01E9H contains a zero byte in the "new" Model 4 ROM only.

0210H-0211H Old: Leftover garbage from the Model I "turn off cassette" routine. New: Two unused zero bytes.

0232H-0234H Old: Leftover garbage from the cassette "blink asterisk" routine. New: Unused zero bytes.

02A8H-02A8H Old: Unused RET instruction. New: Unused zero byte.

02B6H-02B6H Old: Part of LD SP,4288H instruction. New: Part of LD SP,42E8H instruction. This code is used by the BASIC "SYSTEM" command.

02C4H-02C5H Old: Part of JP 06CCH instruction. New: Part of JP 006D instruction. This code is used by the BASIC "SYSTEM" command and fixes a bug. When the BREAK key was pressed in response to the "SYSTEM" command prompt, the "old" ROM would jump to 06CCH which was a "warm re-entry" to BASIC on the Model I, but is part of the LIST/LLIST code on the Model III ROM. The "new" ROM fixes this by jumping to the code that used to be found at 06CCH on the Model I, but is located at 006DH in the "new" ROM.

03CAH-03CAH Part of printer driver - see detailed disassembly.
03CEH-03CEH Part of printer driver - see detailed disassembly.
03D7H-03D8H Part of printer driver - see detailed disassembly.
03E3H-03F9H Part of printer driver - see detailed disassembly.
03FBH-041EH Part of printer driver - see detailed disassembly.
0420H-0449H Part of printer driver - see detailed disassembly.

0469H-046AH Old: Leftover garbage from the Model I video driver routine. New: Two unused zero bytes.

05D1H-05D8H This was a "printer ready" test in the Model I, which was destroyed (for no apparent reason) in the "old" Model III ROM by changing the first three bytes to the characters "RON". In the "new" Model III ROM, the "Cass ?" string literal that was formerly at 37F6H-37FDH has been moved here.

1BC2H-1BC2H In the Model I and in the Models III & 4 using the "old" ROM, memory location 40B0H was used to flag DATA statements while encoding BASIC lines. In the "new" ROM machines memory location 409FH (a previously unused location) is used for this purpose. Thus, this byte is part of a reference to location 40B0H that has been changed to 409FH.

1BDDH-1BDDH Same as above (1BC2H).
1C68H-1C68H Same as above (1BC2H).

2409H-2409H Same as above (1BC2H) ON THE MODEL 4P ONLY. In the "new" Model 4 ROM, this change and the next (2452H) were restored to the previous value of B0H. Why? Only Tandy knows! John Hallgren speculates that the reason may have something to do with the fact that these two are part of arithmetic routines, and the 40B0H location is used to store tokens (IJG's Microsoft BASIC Decoded & Other Mysteries says that the "arith token of last operand (the one to be performed)" is stored here).

2452H-2452H Same as above (2409H).

2FFBH-2FFFH Old: Unused garbage. New: Unused zero bytes.

NEW AND MODIFIED JUMP VECTORS:

3012H Modified vector to disk bootstrap routine (Old: JP 3461H; New: JP 3486H).

3015H Modified vector to reset/power-up routine (Old: JP 3401H; New: JP 3426H).

3024H Modified vector to the keyboard driver routine (Old: JP 338EH; New: JP 3106H)

3027H Modified vector - in the original Model III ROM, this was a jump to the I/O re-router routine at 3739H, in the original Model 4 ROM it contained a RET instruction followed by two NOPs, and in the "new" ROM it contains a jump to the screen print routine at 37A5H.

302DH Modified vector to part of the LIST command (Old: JP 37A4H; New: JP 375CH).

3030H Modified vector to the BASIC TIME$ function routine (Old: JP 37C2H; New: JP 37EAH).

303DH Modified vector to the Non-Maskable Interrupt handler routine (Old: JP 34CEH; New: JP 350EH).

3042H Modified vector to the $SETCAS routine that prompts the user to set the cassette baud rate (Old: JP 310EH; New: JP 33FFH).

3045H NEW vector (JP 378DH) used by line printer driver to translate a character in the range C0H-DFH using the table pointed to by an address stored at 4220H-4221H. This vector is referenced by a CALL at 03F1H if the byte stored at 41FBH = 1.

3048H NEW vector (JP 377AH) used by line printer driver to adjust counters at the start of each new line.

304BH NEW vector (JP 3179H) used to "boot" the computer from the RS-232! There are no references to 304BH in the ROM, however, another reference to 3179H is found at 3518H.

OTHER CODE CHANGES IN THE ROM ABOVE 3000H:

304EH-3085H Keyboard lookup table that formerly started at 3045H has been moved forward nine bytes in the new ROM (locations 3069H through 306DH contain unused zero bytes).

3086H-308DH Part of RS-232 boot routine - see detailed disassembly.

308EH-30C5H Keyboard lookup table that formerly started at 3085H has been moved forward nine bytes in the new ROM (locations 30A9H through 30AEH contain unused zero bytes).

30C6H-30C8H Part of RS-232 boot routine - see detailed disassembly.

30C9H-30CDH A NOP instruction followed by the Floppy Disk Controller time delay routine formerly found at 3518H (in the Model III) or 37E1H (in the "old" Model 4 ROM).

30CEH-3105H Keyboard lookup table that formerly started at 30C5H has been moved forward nine bytes in the new ROM (locations 30E9H through 30EDH contain unused zero bytes).

3106H-3178H Part of keyboard driver routine equivalent to code found at 338EH-3400H in the "old" Model 4 ROM.

3179H-31A4H Part of RS-232 boot routine - see detailed disassembly.

338EH-33F8H Part of keyboard driver routine equivalent to code found at 3739H-37A3H in the "old" Model 4 ROM.

33F9H-3425H Code moved from 3105H-3131H in the "old" Model 4 ROM, includes the $SETCAS routine.

3426H-3516H Equivalent of code found at 3401H-34D9H in the "old" Model 4 ROM (bootstrap routine). Note that the number of retries for a diskette (value at 3487H in the "new" ROM) has been doubled from five to ten).

3517H-3527H Part of RS-232 boot routine - see detailed disassembly.

3739H-375BH Part of keyboard driver routine equivalent to code found at 34DAH-34FCH in the "old" Model 4 ROM.

375CH-3779H Part of BASIC "LIST" command routine moved from 37A4H-37C1H in the "old" ROM.

377AH-37A4H Part of printer driver - see detailed disassembly.

37A5H-37C6H Screen print routine moved from 34FDH-351EH in "old" ROM.

37C7H-37CBH Part of keyboard driver routine equivalent to code found at 313BH-313FH in the "old" Model 4 ROM.

37CCH-37D4H Part of keyboard driver routine equivalent to code found at 37D8H-37E0H in the "old" Model 4 ROM.

37D5H-37E4H Part of RS-232 boot routine - see detailed disassembly.

37EAH-37FCH BASIC TIME$ routine moved from 37C2H-37D7H in "old" ROM.

B. THE NEW PRINTER DRIVER CODE:

```
0043            00100           ORG     0043H
                00110 ;xxxxx CHARACTER >=0E0H & FLAG @ (41FBH)=1  xxxxx
0043 C33404     00120 Q0043H    JP      Q0434H
                00130
0063            00140           ORG     0063H
                00150 ;xxxxx VECTOR TO "NORMAL" CHARACTER PRINT ROUTINE  xxxxx
0063 C31F04     00160 Q0063H    JP      Q041FH
                00170
01DC            00180           ORG     01DCH
01DC CD4B04     00190 Q01DCH    CALL    Q044BH          ;CHECK PRINTER
01DF C9         00200           RET                     ;RETURN IF READYZ
01E0 CD8D02     00210           CALL    Q028DH          ;CHECK <BREAK>
01E3 28F7       00220           JR      Z,Q01DCH        ;LOOP IF NOT PRESSED
01E5 F1         00230           POP     AF              ;EXIT CALLING SUBROUTINE
01E6 C9         00240           RET                     ;RETURN
                00250
028D            00260           ORG     028DH
028D 3A4038     00270 Q028DH    LD      A,(3840H)       ;GET <BREAK> KEY ROW
0290 E604       00280           AND     04H             ;MASK OUT OTHER KEYS
0292 C9         00290           RET                     ;(A=4 IF <BREAK> PRESSED)
                00300
                00310 ;xxxxx ACTUAL START OF PRINTER DRIVER  xxxxx
03C2            00320           ORG     03C2H
03C2 79         00330           LD      A,C             ;PUT CHARACTER IN A
03C3 FE20       00340           CP      20H             ;CONTROL CHARACTER?
03C5 3022       00350           JR      NC,Q03E9H       ;JUMP IF NOT CONTROL
                00360 ;xxxxx CHARACTER IS CONTROL (<20H)  xxxxx
03C7 FE0D       00370           CP      0DH             ;CARRIAGE RETURN?
03C9 2849       00380           JR      Z,Q0414H        ;JUMP IF SO
03CB FE0C       00390           CP      0CH             ;FORM FEED?
03CD 204E       00400           JR      NZ,Q041DH       ;JUMP IF NOT
                00410 ;xxxxx CHARACTER IS FORM FEED  xxxxx
03CF DD7E03     00420           LD      A,(IX+03H)      ;GET # LINES ON PAGE
03D2 DD9604     00430           SUB     (IX+04H)        ;SUBTRACT CURRENT LINE #
03D5 47         00440           LD      B,A             ;B=LINES LEFT ON PAGE
03D6 CDDC01     00450 Q03D6H    CALL    Q01DCH          ;WAIT FOR PRINTER READY
03D9 3E0A       00460           LD      A,0AH           ;LINEFEED CHARACTER IN A
03DB D3F8       00470           OUT     (0F8H),A        ;OUTPUT IT
03DD 10F7       00480           DJNZ    Q03D6H          ;LOOP UNTIL PAGE FEED
03DF DD360508 00490             LD      (IX+05H),08H    ;# CHARACTERS PRINTED = 8
03E3 DD360401 00500             LD      (IX+04H),01H    ;RESET CURRENT LINE COUNT
03E7 185F       00510           JR      Q0443H          ;GO TO EXIT ROUTINE
                00520 ;xxxxx CHARACTER IS NOT CONTROL (<20H)  xxxxx
03E9 3AFB41     00530 Q03E9H    LD      A,(41FBH)       ;GET FLAG
03EC B7         00540           OR      A               ;IS IT ZERO?
03ED 2807       00550           JR      Z,Q03F6H        ;GO IF SO
03EF FE01       00560           CP      01H             ;FLAG=1?
03F1 CA4530     00570           JP      Z,Q3045H        ;GO IF SO
03F4 1829       00580           JR      Q041FH          ;GO IF FLAG NOT 0 OR 1
                00590 ;xxxxx CHARACTER NOT CONTROL & FLAG @ (41FBH)=0  xxxxx
03F6 3AFC41     00600 Q03F6H    LD      A,(41FCH)       ;GET FLAG
03F9 B7         00610           OR      A               ;IS IT ZERO?
03FA 200E       00620           JR      NZ,Q040AH       ;GO IF NOT
                00630 ;xxxxx CHAR NOT CTRL & FLAGS @(41FBH) & (41FCH)=0  xxxxx
03FC 79         00640           LD      A,C             ;GET CHARACTER
03FD FEA0       00650           CP      0A0H            ;IS CHARACTER <0A0H?
03FF 381E       00660           JR      C,Q041FH        ;GO IF SO
0401 FEC0       00670           CP      0C0H            ;IS CHARACTER <0C0H?
```

```
0403 300A    00680          JR     NC,0040FH      ;GO IF NOT
             00690 ;CHARACTER IN RANGE A0H - BFH
0405 C640    00700          ADD    A,40H          ;MAKE CHARACTER E0H-FFH
0407 4F      00710          LD     C,A            ;SAVE IT
0408 1815    00720          JR     0041FH         ;GO OUTPUT IT
             00730 ;xxxxx CHAR NOT CTRL, FLAG @(41FBH)=0,@(41FCH)<>0  xxxxx
040A 79      00740 0040AH   LD     A,C            ;GET CHARACTER
040B FEC0    00750          CP     0C0H           ;IS CHARACTER <0C0H?
040D 3810    00760          JR     C,0041FH       ;GO IF SO
             00770 ;CHARACTER IN RANGE C0H - FFH
040F D620    00780 0040FH   SUB    20H            ;MAKE CHARACTER A0H-DFH
0411 4F      00790          LD     C,A            ;SAVE IT
0412 1808    00800          JR     0041FH         ;GO OUTPUT IT
             00810 ;xxxxx CHARACTER IS CARRIAGE RETURN  xxxxx
0414 DD7E05  00820 00414H   LD     A,(IX+05H)     ;GET # CHRS PRINTED ON LN
0417 B7      00830          OR     A              ;ANY CHARS ON THIS LINE?
0418 201A    00840          JR     NZ,00434H      ;GO IF SO
041A 3E0A    00850          LD     A,0AH          ;CONVERT <CR> TO <LF>
041C 4F      00860          LD     C,A            ;STORE LINEFEED CHARACTER
             00870 ;xxxxx JUMP HERE IF CHAR CTRL BUT NOT <CR> OR <LF> xxxxx
041D 1815    00880 0041DH   JR     00434H
             00890 ;xxxxx THIS ROUTINE PRINTS "NORMAL" CHARACTERS  xxxxx
041F DD7E06  00900 0041FH   LD     A,(IX+06H)     ;GET MAXIMUM LINE LENGTH
0422 3C      00910          INC    A              ;IF NO MAXIMUM, NOW A=0
0423 280F    00920          JR     Z,00434H       ;GO IF NO MAXIMUM LENGTH
0425 DD8E05  00930          CP     (IX+05H)       ;MAXIMUM LENGTH REACHED?
0428 300A    00940          JR     NC,00434H      ;GO IF NOT
042A CD0C01  00950          CALL   001DCH         ;WAIT FOR PRINTER READY
042D 3E0D    00960          LD     A,0DH          ;CARRIAGE RETURN IN A
042F D3F8    00970          OUT    (0F8H),A       ;OUTPUT IT
0431 CD4830  00980          CALL   03048H         ;ADJUST COUNTERS
0434 CD0C01  00990 00434H   CALL   001DCH         ;WAIT FOR PRINTER READY
0437 79      01000          LD     A,C            ;GET CHARACTER TO PRINT
0438 D3F8    01010          OUT    (0F8H),A       ;OUTPUT IT TO PRINTER
043A DD3405  01020          INC    (IX+05H)       ;INCREMENT # CHRS PRINTED
043D FE0D    01030          CP     0DH            ;CARRIAGE RETURN?
043F 2804    01040          JR     Z,00445H       ;FIX COUNTERS IF SO
0441 FE0A    01050          CP     0AH            ;LINEFEED?
0443 2003    01060          JR     NZ,00448H      ;SKIP IF NOT
0445 CD4830  01070 00445H   CALL   03048H         ;ADJUST COUNTERS
0448 AF      01080 00448H   XOR    A              ;CLEAR STATUS
0449 79      01090          LD     A,C            ;RESTORE CHARACTER
044A C9      01100          RET                   ;DONE
044B DBF8    01110 0044BH   IN     A,(0F8H)       ;GET PRINTER STATUS
044D E6F0    01120          AND    0F0H           ;MASK OFF IRRELEVENT BITS
044F FE30    01130          CP     30H            ;READY?
0451 C9      01140          RET
             01150
3045         01160          ORG    03045H
3045 C38D37  01170 03045H   JP     0378DH         ;CHR NOT CTL, (41FBH)=1
3048 C37A37  01180 03048H   JP     0377AH         ;ADJUST COUNTERS
             01190
377A         01200          ORG    0377AH
377A DD360500 01210 0377AH  LD     (IX+05H),00H   ;RESET # CHRS PRNTD ON LN
377E DD3404  01220          INC    (IX+04H)       ;INCREMENT # LINES PRNTED
3781 DD7E04  01230          LD     A,(IX+04H)     ;GET NEW LINE COUNT
3784 DD8E03  01240          CP     (IX+03H)       ;ON NEXT PAGE?
3787 C0      01250          RET    NZ             ;RETURN IF NOT
3788 DD360401 01260         LD     (IX+04H),01H   ;RESET LINE COUNT TO TOP
378C C9      01270          RET                   ;  OF PAGE & RETURN
             01280 ;xxxxx CHARACTER NOT CONTROL & FLAG @ (41FBH)=1  xxxxx
378D 79      01290 0378DH   LD     A,C            ;GET CHARACTER
378E FEE0    01300          CP     0E0H           ;CHARACTER <0E0H?
3790 D24300  01310          JP     NC,00043H      ;GO IF NOT (TO 0434H)
3793 FEC0    01320          CP     0C0H           ;CHARACTER <0C0H?
3795 DA6300  01330          JP     C,00063H       ;GO IF SO (TO 041FH)
             01340 ;CHARACTER IN RANGE 0C0H-0DFH
3798 D6C0    01350          SUB    0C0H           ;MAKE CHARACTER 00H-1FH
379A 0600    01360          LD     B,00H          ;BC=ADJUSTED CHARACTER
379C 4F      01370          LD     C,A            ;  (IN RANGE 0000H-001FH)
3790 2A2042  01380          LD     HL,(4220H)     ;GET ADDR OF LOOKUP TABLE
37A0 09      01390          ADD    HL,BC          ;ADD OFFSET
37A1 4E      01400          LD     C,(HL)         ;GET CHAR FROM LOOKUP TBL
37A2 C36300  01410          JP     00063H         ;GO TO VECTOR TO 041FH
             01420
0000         01430          END
00000 TOTAL ERRORS
```

Memory locations 41FBH, 41FCH, and 4220H-4221H are used for the first time (by the printer driver) in the above code. I don't know why the changes were made, but the net effect is as follows:

1) If memory location 41FBH contains any value other than 0 or 1, all characters are handled normally. Unfortunately, this location is initialized with a value of zero. So if your graphics printer starts doing funny things when connected to a TRS-80 Model 4 or 4P with the new ROM (or ROM image), simply try POKEing memory location 41FBH with any number between 2 and 255.

2) If memory location 41FBH contains a value of 1, then characters in the range C0H through DFH will be converted according to the contents of a 20H byte long table beginning at an address stored at 4220H-4221H (these locations contain an address of 0000H at initialization, which obviously is not pointing at a lookup table!). If you want to use this feature from BASIC, you could create a 32 (decimal) byte string literal (for example, A$="... (32 bytes) ..."), then use the VARPTR function to find the start of the string in memory and POKE that into locations 4220H-4221H. Don't forget to poke a value of 1 into 41FBH. After that, any byte you send to the printer from C0H through DFH should be converted according to the characters in your string literal (C0H would be converted to the first character, C1H to the second, and so on...).

3) If memory location 41FBH contains a value of 0 (as it does at initialization), then characters in the range C0H through FFH will have 20H subtracted from their value (they will be converted to characters in the range A0H through DFH). IN ADDITION, if (and only if) memory location 41FCH contains a value of 0 (as it does at initialization), then characters in the range A0H through BFH will have 40H ADDED to their value (they will be converted to characters in the range E0H through FFH).

The following chart may illustrate this a bit better:

Value in (41FBH)	Value in (41FCH)	Original character value (offset shown below) A0H-BFH	C0H-DFH	E0H-FFH
2-255	don't care	no change	no change	no change
1	don't care	no change	table lookup	no change
0	1-255	no change	20H subtracted	20H subtracted
0	0	40H added	20H subtracted	20H subtracted

C. THE NEW RS-232 BOOT ROUTINE CODE:

```
3040         00100          ORG    3040H
3040 C37931  00110 03040H   JP     03179H         ;VECTOR TO RS-232 BOOT
             00120
             00130 ;$$$$$ GET CHARACTER FROM RS-232
3086         00140          ORG    3086H
3086 DBEA    00150 03086H   IN     A,(0EAH)       ;GET UART STATUS
3088 B7      00160          OR     A              ;DATA RECEIVED IN UART?
3089 F28630  00170          JP     P,03086H       ;IF NOT, TRY AGAIN
308C 1838    00180          JR     030C6H         ;ELSE GET CHAR FROM UART
             00190
30C6         00200          ORG    30C6H
30C6 DBE8    00210 030C6H   IN     A,(0E8H)       ;GET CHAR FROM UART
30C8 C9      00220          RET                   ;  & RETURN
             00230
             00240 ;$$$$$ ACTUAL START OF RS-232 BOOT ROUTINE  $$$$$
3179         00250          ORG    3179H
3179 AF      00260 03179H   XOR    A              ;A=0
317A D3E8    00270          OUT    (0E8H),A       ;MASTER RESET OF UART
317C 3EEE    00280          LD     A,0EEH         ;SELECT 9600 BAUD FOR
317E D3E9    00290          OUT    (0E9H),A       ;  INPUT/OUTPUT
3180 3E6D    00300          LD     A,6DH          ;INIT UART ODD,8BIT,1STOP
3182 D3EA    00310          OUT    (0EAH),A       ;  NO PRTY;RTS,BRK,DTR OFF
3184 DBE8    00320 03184H   IN     A,(0E8H)       ;GET MODEM STATUS REGISTR
3186 CB77    00330          BIT    6,A            ;"DSR" ON?
3188 28FA    00340          JR     Z,03184H       ;IF NOT, KEEP WAITING
318A 3E6C    00350          LD     A,6CH          ;INITIALIZE UART AGAIN
318C D3EA    00360          OUT    (0EAH),A       ;  WITH DTR ON
318E DBE8    00370 0318EH   IN     A,(0E8H)       ;GET MODEM STATUS REGISTR
3190 CB77    00380          BIT    6,A            ;"DSR" ON?
3192 20FA    00390          JR     NZ,0318EH      ;IF SO, WAIT FOR OFF
3194 3E0F    00400          LD     A,0FH          ;CHAR TO TRANSMIT = <SI>
3196 CD0537  00410          CALL   037D5H         ;OUTPUT CHAR TO RS-232
3199 CD8630  00420          CALL   03086H         ;GET A CHARACTER
319C CD8630  00430          CALL   03086H         ;GET ANOTHER CHARACTER &
319F CD0537  00440          CALL   037D5H         ;  OUTPUT IT TO RS-232
31A2 C31735  00450          JP     03517H         ;CONTINUE IN ROUTINE
             00460
3517         00470          ORG    3517H
3517 B7      00480 03517H   OR     A              ;WAS 2ND CHARACTER A 0?
3518 C27931  00490          JP     NZ,03179H      ;IF NOT, ABORT & RETRY
```

```
3518 210043   00580          LD     HL,4300H      ;4300H=NORMAL BOOT BUFFER
351B CD8630   00510 0351EH   CALL   0308AH        ;GET A CHARACTER
3521 77       00520          LD     (HL),A        ;SAVE IT TO BUFFER
3522 2C       00530          INC    L             ;INCREMENT BUFFER POINTER
3523 20F9     00540          JR     NZ,0351EH     ;BACK TO 4300? NO,DO NEXT
3525 C3E037   00550          JP     037E0H        ;GO TO EXIT ROUTINE
              00560
 7D5          00570          ORG    37D5H
 /D5 F5       00580 037D5H   PUSH   AF            ;SAVE CHARACTER & STATUS
37D6 DBEA     00590 037D6H   IN     A,(0EAH)      ;GET UART STATUS
37D8 CB77     00600          BIT    6,A           ;ANY CHARACTER WAITING?
37DA 28FA     00610          JR     Z,037D6H      ;IF NOT, TRY AGAIN
37DC F1       00620          POP    AF            ;RELOAD CHARACTER
37DD D3EB     00630          OUT    (0EBH),A      ;GIVE TO RS-232
37DF C9       00640          RET                  ; & RETURN
37E0 3E60     00650 037E0H   LD     A,60H         ;SET "DTR" BACK ON
37E2 D3EA     00660          OUT    (0EAH),A      ;  AGAIN
37E4 E9       00670          JP     (HL)          ;JUMP TO 4300H
              00680
3046          00690          END    03046H
00000 TOTAL ERRORS
```

Quite frankly, I'm not absolutely certain of the purpose of the above code, but I believe it was designed to allow the 4P to be booted off the RS-232 interface in a "networking" type situation. As most 4P owners know, holding down certain keys while booting up the 4P will select different modes of operation (see "MODEL 4P BOOT MODE KEY SELECTION" in NORTHERN BYTES Volume 5, Number 4, page 15). One of the "undocumented" combinations is to hold down the SHIFT and BREAK keys while booting up, which supposedly will boot from the RS-232. This is most likely the code that handles that.

D. CONCLUSION

If my mail is any indication, the most significant change in the "new" ROM (the one that seems to be giving folks the most fits) is the revised printer driver. So remember, if you have a program that prints properly when using a Model III (or an early Model 4), but doesn't work correctly when using a "new" ROM Model 4 or 4P, go to BASIC and type:

POKE 16891,2 .

as should restore normal operation. Since this memory location (41FBH) is unused under the "old" ROM, you may wish to add this instruction near the beginning of your Model III/4 BASIC programs that send graphics characters to the printer.

Once again, I wish to give special thanks to John Hallgren (1939 Atlantis Drive, Clearwater, Florida 33575) for providing the bulk of the above information, and for proofreading and debugging a first draft of this article. John has also sent me a hardcopy listing of a fairly well-commented disassembly of the Model 4P boot ROM. Due to the length of this disassembly (this is 4K of code we're talking about!) and the fact that the routines in the boot ROM cannot normally be accessed from within a running program anyway (well, they CAN, but most folks would have no reason to), I don't think I will print it in Northern Bytes (if I get a flood of mail protesting this decision, I may change my mind). I only mention this to let you know that someone HAS disassembled the 4P boot ROM.

THE PROTECTION SCHEME OF COPYCAT 3

by Mohammad Dadashzadeh, Ph.D.

[This article is a condensed excerpt from my forthcoming book - COPY PROTECTION SCHEMES ON THE TRS-80. The book is subtitled, how to copy that protected disk, as well as how to convert that protected tape (disk) into a /CMD file.]

The protection scheme of COPYCAT 3 falls under the class which I have named NODAM. Recall that diskettes are formatted by constructing a formatting data record in memory and then issuing a write track command. The formatting data record includes not only the initial data for the sectors defined for that track but also sector identification records as well as inter-sector filler blocks. A sector identification record consists of four bytes: track number, side number, sector number, and length code. It must be preceded by an ID ADDRESS MARK, i.e., the byte FE, and immediately followed by the byte F7 which instructs the floppy disk controller (FDC) to write the two-byte Cyclic Redundancy Checksum associated with the previous record. For synchronization purposes, a filler block must be used to separate the sector identification record from the sector data record the start of which is signified to the FDC with a DATA ADDRESS MARK (DAM), which is one of the bytes F8, F9, FA, or FB. The sector data record must also be immediately followed by the byte F7.

The NODAM protection scheme is based on constructing a formatting data record with a single sector identification record, and following it with a filler block that is in fact the protected information. In other words, there will be NO Data Address Mark. As such, copying programs that are based on copying sectors are defeated.

To defeat the NODAM protection scheme, it is sufficient to read the entire track using the read track command while synchronizing on the lone ID ADDRESS MARK, locating the sector identification record which is signified by the byte FE, replacing the two-byte CRC with the byte F7, and writing this formatting data record to the destination diskette. Unfortunately, there is a problem which prevents the complete automation of this procedure. Simply stated, on a track read it is possible that many incorrect FE bytes will be produced. And because the sector identification record may conatin false CRC values, there is no algorithmic way to determine the start of the filler block that is the protected information.

On the other hand, the same problem is faced by the loader of the protected information. The simplest solution is to start the filler block with a code sequence. Although it is possible to make an educated guess at the code sequence by examining a few such protected tracks, the certain way is through boot tracking. This is why the breaking of NODAM protected diskettes has remained in the domain of a select few.

The earliest NODAM protection scheme (known to me) is the wonderful work of Yves Lempereur of FUNSOFT games. He also introduced the reverse nibble encryption scheme (e.g., CD is encoded as 0D 0C) which is used by COPYCAT 3.

Below is a summary guide for backing up COPYCAT 3 (a Model 3 must be used). The full guide and complete instructions on converting it to a /CMD file appear elsewhere in the book.

(All numbers are in hexadecimal. SU+ is Kim Watt's Super Utility Plus. And, don't forget to write protect your COPYCAT 3.)

Step 1 - Use the special backup function of SU+ to backup COPYCAT 3. Essentially all that is needed is to copy tracks 0 and 13 of COPYCAT 3 which are unprotected.

On track 0, there are 4 sectors with the following sector identification records:

Track #: 00	Side #: 00	Sector #: 01	Length Code: 01
Track #: 69	Side #: 00	Sector #: 69	Length Code: 03
Track #: 40	Side #: 00	Sector #: 40	Length Code: 03
Track #: 30	Side #: 00	Sector #: 30	Length Code: 00

On track 13, there are 2 sectors with the following sector identification records:

Track #: 30	Side #: 00	Sector #: 30	Length Code: 03
Track #: 40	Side #: 00	Sector #: 40	Length Code: 03

Repeat step 2 for each protected track of COPYCAT 3. They are tracks 2 through 8, each with the following sector identification record:

Track #: 31	Side #: 16	Sector #: 31	Length Code: 00

Step 2 - Use the track to memory function of SU+ to read the protected track of COPYCAT 3. Once the protected track is in memory, locate the code sequence used by COPYCAT 3. That is, look for 4C 4E (L N in ASCII, which are, by the way, the initials of the author of COPYCAT 3). You can actually expect to see the sequence: FE 31 16 31 00 44 26 4C 4E. Modify the bytes before 4C 4E so that they read: F6 F6 F6 FC 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 00 00 F5 F5 F5 FE 31 16 31 00 F7 4C 4E. You may also wish to replace the bytes preceding the first F6 with 4E. Next, use the memory to track function of SU+ to write this formatting data record to the destination diskette.

HELP WANTED - CAN YOU HELP?

A gentleman here in Sault Ste. Marie is still using Radio Shack's MICROFILES program. Unfortunately, his young daughter took a black magic marker to many of the pages, and ... you can guess the rest. If you have a MICROFILES manual you're not using any more, we can give it a good home!

FKEY/CMD
A public domain program from the
Christian Computer Users Association TRS-80 Library
Provided by Bob Grommes
1733 Eastern S.E., Grand Rapids, Michigan 49507

This machine-language utility runs under Model 4 TRSDOS Version 6 and allows you to change the code values generated by the three function keys. It will redefine any or all of the function keys, both shifted and unshifted, and/or display the current settings of the keys. This has many practical uses in any program that uses the standard keyboard device (*KI) to scan the keyboard; for example, when writing BASIC programs, if you make much use of the carat symbol (used for exponentiation), you will find it easier to define a function key to that particular symbol rather than to constantly press <CLEAR> and <↑> at the same time. Commonly used Disk Scripsit control keys such as <CLEAR> <S> could be generated by a single function key, often saving one or two keystrokes, as well as being less awkward besides.

The command syntax for FKEY is as follows:

FKEY

. . . without any parameters will simply display the current settings of the unshifted function keys.

```
FKEY (F1="[",F2="]",F3="^",S1="{",S2="}",S3="|")
FKEY (F1=91,F2=93,F3=94,S1=123,S2=125,S3=124)
FKEY (F1=X'5B',F2=X'5D',F3=X'5E',S1=X'7B',S2=X'7D',S3=X'7C')
```

. . . the above three commands produce identical results. The F1, F2 and F3 parameters define the unshifted function keys; the S1, S2 and S3 parameters define the shifted function keys (that is, <SHIFT> <F1>, <SHIFT> <F2>, etc.). Each parameter can be defined as a quoted string, a decimal value or a hex value in X'nn' format. You can mix any combination of string, decimal or hex values on a command line, and the order of the parameters isn't important. Of course, you don't have to use all of the parameters either, only those you want to.

FKEY (DEFAULT)

This command will reset the function keys to their default values.

Since FKEY executes entirely within the library overlay memory region, it is callable from any program that allows you to invoke TRSDOS library commands. Often, you will have to do this by using the TRSDOS command RUN. For example, you can run FKEY from BASIC with the following command:

SYSTEM "RUN FKEY (parameters)"

Please note that you cannot use FKEY to allow a function key to emulate the SHIFT - @ function of PAUSE. Since TRSDOS specifically scans SHIFT and @ to detect the PAUSE condition, even though a function key may generate the code value of X'60', it will NOT be detected as PAUSE.

[The following additional comments on FKEY/ASM were provided by Bob Grommes.]

Just a few technical notes on FKEY and TRSDOS 6 assembly language coding in general:

The whole subject of Supervisor Calls (SVC's) used in TRSDOS 6 and other Model 4 DOS's may be a little unfamiliar to those of us accustomed to Model I/III assembly language practices. FKEY, like almost all machine code for the native Model 4 mode, makes extensive use of SVC's. Fortunately they are quite simple to understand and have some real advantages to the programmer. Consider the following code:

Model III:

```
GET     EQU     013H

        LD      DE,FCB
        CALL    GET
```

Model 4:

```
@GET    EQU     3

        LD      A,@GET
        LD      DE,FCB
        RST     40
```

Both of these routines input a single byte from a device or file pointed to by DE. Both take up 6 bytes of memory, and both take 8 machine cycles to execute. Why use a SVC then? Well, it allows routines within the operating system to "have legs", and move around freely during different releases of the DOS, without us programmers having to chase them around. The bad news is that all user SVC's in TRSDOS 6 use the RST 40 (or RST 28H) instruction. This means there is not a lean, mean subroutine sitting right at 28H. It means there's a jump vector to a routine that looks at th accumulator, sees what SVC you want, accesses another table of vectors and THEN jumps to the appropriate subroutine. This means more overhead than some CALLs. On the other hand, many routines in the familiar Model III ROM were accessed via vectors and used various pointers to pointers to pointers also. Some also needed fairly involved stack setup to properly invoke as well. So the extra overhead may not be that much of a real problem when you look at the big picture and the Model 4's faster clock speed. In any case, despite the relative merits of SVC's, we have to deal with them, because they're there!

The SVC's used in FKEY have the following functions:

```
@DSPLY  Display a message string
@PARAM  Parse the command line for any parameters in parentheses
@GTMOD  Get the address of a device driver given the device name
@HEX8   Convert a one-byte value to ASCII Hex representation
@LOGOT  Same as @DSPLY except the message is also sent to the
        JOBLOG, if active.
```

@DSPLY should be very easy to grasp as it's just a familiar "display a string pointed to by HL" routine. @LOGGER, not used in this program, does the same thing except the message goes to the joblog, if it's active. @LOGOT, which IS used here, is the same as using @DSPLY followed by @LOGGER. You should use @LOGOT instead of @DSPLY for error messages, because if the user had joblog active, he will want any error messages to show up there as well as on the display.

@PARAM, the parameter scanner, is a real code saver since it will do almost any command line parsing you can imagine with a single call. The only overhead is your parameter table (represented in FKEY by PRMTBL$) which tells @PARAM what your valid parameters are and whether or not abbreviations are legal, etc. A detailed explanation of @PARAM is beyond the scope of this articl but it's well worth your taking the time to study the Technica. Reference Manual, which shows how a parameter table is set up. It's not as exotic as FKEY makes it look, it's just that the programmer here was using some of his own personal short-hand short-cuts and the features of the EDAS assembler package to produce the table.

Anyway, the theory behind FKEY is simple: First, find the data area of the keyboard driver, a task easily accomplished with @GTMOD. The DE register is pointed to the two-character module name KI. After issuing the @GTMOD SVC, HL will contain the entry address of the module, which we throw away because we don't need it. DE, on the other hand, will point to the start of the "data area" of the *KI driver. Just what is it that we find here? According to what documentation I have, it looks like this:

```
DE ==>      the last character entered
DE+1 ==>    Contains the repeat time check which is the system's
            timer value that when reached will result in a repeat of
            the last character if the keycode scanned has not
            changed.
DE+2 ==>    Contains the waiting time in timer units that must
            transpire before a character can initially be repeated.
            This value is set by the SETKI (Wait=nn) library
            command.
DE+3 ==>    Contains the repeat rate in timer units. Set by SETKI
            (Rate=nn).
DE+32 ==>   The function key table (on Model 4 and 4P only, may be
            elsewhere on the MAX 80 and other machines on which
            TRSDOS 6 is implemented).
```

So at this point, FKEY just adds 32 to DE and stuffs whatever value(s) you supply on the command line into the function key code table.

An interesting point is borne out here for anyone using th Model 4 function keys. Since it's possible for these keys to be redefined, you may want to add initialization code to any program that uses these keys which will save whatever is in the function key table, insert whatever values your program expects (even the default values) and then, upon termination of your program, restores whatever was in the function key table at entry. This

would prevent a user from inadvertently locking himself out of certain functions of your program or even causing unexpected things to happen when pressing the function keys!

```
                ;FKEY/ASM - 05/03/84
                ;=-=-=
                ;Program to set function key codes in TRSDOS 6.x
                ;       Use: FKEY (DEFAULT,F1=,S1=,...
                ;       Sn = shifted Fn. Formats: Fn="c"; Fn=ddd; Fn=x'hh'
                ;       Released to public domain by Roy Soltoff - 04/13/84
                ;=-=-=
0000            F1      EQU     0
0001            S1      EQU     1
0002            F2      EQU     2
0003            S2      EQU     3
0004            F3      EQU     4
0005            S3      EQU     5
2600                    ORG     2600H
2600 E5         FKEY    PUSH    HL              ;Save command pointer
2601 21F626             LD      HL,HELLO$
2604 3E0A               LD      A,10            ;@DSPLY
2606 EF                 RST     40
2607 E1                 POP     HL
2608 118326             LD      DE,PRMTBL$      ;Get parameter values
260B 3E11               LD      A,17            ;@PARAM
260D EF                 RST     40
260E C2A926             JP      NZ,PRMERR
2611 116227             LD      DE,KIMOD$       ;Locate *KI driver data area
2614 3E53               LD      A,83            ;@GTMOD
2616 EF                 RST     40
2617 C2A526             JP      NZ,KIERR
261A 212000             LD      HL,32           ;Index to function key table
261D 19                 ADD     HL,DE
261E E5                 PUSH    HL              ;Save on stack
261F DD218726           LD      IX,PRMTBL$+4    ;Test if user wants to
2623 DD7E00             LD      A,(IX+F1%6)     ;  default to system values
2626 DDB606             OR      (IX+S1%6)
2629 DDB60C             OR      (IX+F2%6)
262C DDB612             OR      (IX+S2%6)
262F DDB618             OR      (IX+F3%6)
2632 DDB61E             OR      (IX+S3%6)
2635 2037               JR      NZ,CHGKEY
2637 010000     DPARM   LD      BC,0            ;Init DEFAULT=OFF
263A B0                 OR      B
263B B1                 OR      C
263C 2830               JR      Z,CHGKEY        ;Go to p/u current key settings
263E 21E426     MOVKEYS LD      HL,DEFKEY       ;Move new function key
2641 D1                 POP     DE              ;  code table
2642 D5                 PUSH    DE              ;Save pointer to table
2643 010600     MOV6    LD      BC,6
2646 EDB0               LDIR
2648 D1                 POP     DE              ;Recover table pointer
2649 21BE27             LD      HL,KEYS$-1      ;Index to start of buffer
264C 0603               LD      B,3             ;Loop for 3 keys
264E C5         DSPLP   PUSH    BC
264F 010600             LD      BC,6
2652 09                 ADD     HL,BC           ;Point to next field
2653 CD6626             CALL    GETKEY          ;Unpack unshifted
2656 CD6626             CALL    GETKEY          ;Unpack shifted
2659 C1                 POP     BC
265A 10F2               DJNZ    DSPLP
265C 218F27             LD      HL,KEYS$
265F 3E0A               LD      A,10            ;@DSPLY
2661 EF                 RST     40
2662 210000     EXIT    LD      HL,0
2665 C9                 RET
2666 1A         GETKEY  LD      A,(DE)          ;P/u key code
2667 13                 INC     DE
2668 4F                 LD      C,A
2669 3E62               LD      A,98            ;@HEX$
266B EF                 RST     40
266C 23                 INC     HL
266D C9                 RET
266E 11E426     CHGKEY  LD      DE,DEFKEY
2671 E1                 POP     HL              ;Point to KI's table
2672 E5                 PUSH    HL              ;Save for update
2673 0606               LD      B,6             ;Init for six tests
2675 11E326             LD      DE,DEFKEY-1     ;Point to start of table
2678 CD7F26     KEYLP   CALL    PARSE
267B 10FB               DJNZ    KEYLP
267D 188F               JR      MOVKEYS
267F DD7E00     PARSE   LD      A,(IX)          ;P/u response code
2682 DD23               INC     IX
2684 DD6E00             LD      L,(IX)          ;P/u vector
2687 DD23               INC     IX
2689 DD6600             LD      H,(IX)
268C DD23               INC     IX
268E DD23               INC     IX
2690 DD23               INC     IX
2692 DD23               INC     IX
2694 13                 INC     DE              ;Bump pointer to table
2695 B7                 OR      A
2696 C8                 RET     Z
2697 CB7F               BIT     7,A
2699 2007               JR      NZ,PARSE2
269B CB6F               BIT     5,A
269D C8                 RET     Z
269E 7E                 LD      A,(HL)
269F 23                 INC     HL
26A0 66                 LD      H,(HL)
26A1 6F                 LD      L,A
26A2 7E         PARSE2  LD      A,(HL)          ;Get the new value
26A3 12                 LD      (DE),A          ;Replace with new value
26A4 C9                 RET
26A5 217727     KIERR   LD      HL,KIERR$
26A8 DD         PRMERR  DB      0DDH
26A9 216627             LD      HL,PRMERR$
26AC 3E0C               LD      A,12            ;@LOGOUT
26AE EF                 RST     40
26AF 21FFFF             LD      HL,-1           ;Init for error
26B2 C9                 RET
26B3 80         PRMTBL$ DB      80H
26B4 A2463100           DB      0A2H,'F1',0
26B8 EA26               DW      F1%2+PARMS
26BA A2533100           DB      0A2H,'S1',0
26BE EC26               DW      S1%2+PARMS
26C0 A2463200           DB      0A2H,'F2',0
26C4 EE26               DW      F2%2+PARMS
26C6 A2533200           DB      0A2H,'S2',0
26CA F026               DW      S2%2+PARMS
26CC A2463300           DB      0A2H,'F3',0
26D0 F226               DW      F3%2+PARMS
26D2 A2533300           DB      0A2H,'S3',0
26D6 F426               DW      S3%2+PARMS
26D8 57444546           DB      57H,'DEFAULT',0
26DC 41 55 4C 54 00
26E1 3826               DW      DPARM+1
26E3 00                 NOP
26E4 81918292   DEFKEY  DB      81H,91H,82H,92H,83H,93H
26E8 83 93
26EA 00000000   PARMS   DW      0,0,0,0,0,0
26EE 00 00 00 00 00 00 00 00
26F6 0A4648454C HELLO$  DB      10,'FKEY - Change or reset Model 4 function keys'
26FA 59 20 2D 20 43 68 61 6E 67 65 20 6F 72 20 72 65 73 65 74 20
270E 4D 6F 64 65 6C 20 34 20 66 75 6E 63 74 69 6F 6E 20 68 65 79
2722 73
2723 0A507562           DB      10,'Public Domain - Written by Roy Soltoff'
2727 6C 69 63 20 44 6F 6D 61 69 6E 20 2D 20 57 72 69 74 74 65 6E
2738 20 62 79 20 52 6F 79 20 53 6F 6C 74 6F 66 66
274A 2C20416C           DB      ', All rights reserved.',10,13
274E 6C 20 72 69 67 68 74 73 20 72 65 73 65 72 76 65 64 2E 0A 0D
2762 244B4D4F3  KIMOD$  DB      '*KI',3
2766 50617261   PRMERR$ DB      'Parameter error!',13
276A 6D 65 74 65 72 20 65 72 72 6F 72 21 0D
2777 43616E27   KIERR$  DB      'Can',39,'t locate *KI driver',13
277B 74 20 6C 6F 63 61 74 65 20 2A 4B 49 20 64 72 69 76 65 72 0D
278F 46312078   KEYS$   DB      'F1 = xx,yy; F2 = xx,yy; F3 = xx,yy',13
27A3 20 78 78 2C 79 79 3B 20 46 32 20 3D 20 78 78 2C 79 79 3B 20
27B7 46 33 20 3D 20 78 78 2C 79 79 0D
2600                    END     FKEY
```

SYMBOL TABLE

F1	0000	S1	0001	F2	0002	S2	0003	F3	0004
S3	0005	FKEY	2600	DPARM	2637	MOVKEYS	263E	MOV6	2643
DSPLP	264E	EXIT	2662	GETKEY	2666	CHGKEY	266E	KEYLP	2678
PARSE	267F	PARSE2	26A2	KIERR	26A5	PRMERR	26A9	PRMTBL$	26B3
DEFKEY	26E4	PARMS	26EA	HELLO$	26F6	KIMOD$	2762	PRMERR$	2766
KIERR$	2777	KEYS$	278F						

MAGIC MATH PLUS
A continuing series feature in Northern Bytes
by Dr. Michael W. Ecker

[Dr. Michael Ecker, an Associate Professor of Mathematics and Computer Science at the University of Scranton, is a contributing editor of Popular Computing and Soft Sector (and formerly of Byte and Computer Gaming World) for which computer magazines he writes columns on mathematical and computer recreations. Most notable is his monthly column "Recreational Computing" in Popular Computing. A book based on material similar to that in his columns is scheduled for release by Arco Publishing in the Fall of 1985. He also does occasional freelance pieces on financial mathematics, and software reviews. As the alter ego of Recreational Mathemagical Software, Dr. Ecker offers this serialization of some of his Magic Math Plus software, a collection of programs of "mathemagic", games, educational programs, financial utilities and numerous number tricks. At least one program from Magic Math will b offered in each installment. Readers who would prefer not to type in programs can purchase a disk directly from him. Northern Bytes is pleased to extend its coverage of useful information for TRS-80s to basic Basic applications and recreations. Information regarding questions for Dr. Ecker and/or ordering will be provided at the end of each article.]

Superblackjack: The Game of 110
by Michael W. Ecker, PhD

Mathematicians are never content, it seems, to leave well enough alone. After enjoying blackjack, perhaps you too could use a change. With that in mind, consider this mathematical variation of that game.

In 110, you play against your computer. Your goal is to get a score as close to 110 as possible without going over. In fact, it has to be closer to 110 than your opponent's. If all your possible scores go over 110, you lose immediately. All ties will be considered a draw. There may be no betting, insurance, doubling, surrendering or splitting pairs. The computer will keep track of the number of games won. Unlike 21's rules, all 110s are considered equal, regardless of whether 110 arises as a super-blackjack (ace and ten value) or by more than two cards. Note that 10x11 = 110, as opposed to 10+11 = 21; i.e. there is some multiplication involved.

As with 21, you are dealt cards and keep track of your card point totals. However, the way these are calculated is different here. Indeed, the most important difference between 110 and 21 is that here you multiply and sometimes add, instead of just adding. You are each dealt two cards. Both of yours are face up, as are the computer's. (In 21, you would not see the dealer's second card. But it's your computer, so why should you give it any advantages?) Multiply your two cards' face values, counting picture cards as 10. An ace on the first two cards (of player or dealer) may be counted as 1 or 11. (Due to programming difficulties, later aces are counted solely as 11.) A 110 (an ace and either a ten or a picture card), is analogous to 21 - blackjack. (Get it?) Suppose you have an ace and a seven, giving you a 1 x 7 = 7 or 11 x 7 = 77 so far. Hmmm... not very close to 110... You may wish to take another card. Whatever you draw next you add to 7 or 77. So, if you now get a picture card (11, 12, 13) or ten, your score is now 7 + 10 = 17 or 77 + 10 = 87. Now here is the tricky part. If you decide to take still another card in order to get closer to 110, the previous card - having a value of 10 in this case - is no longer counted as before. Say your fourth card is a 3. You instead multiply your third and fourth cards' values together: 10 x 3 = 30. Now add this product to the previous one(s): 7 + 30 = 37, or 77 + 30 = 103. If you wish to take one more card, its value is added to your 37 or 103. This process continues in the same way. Note that only first or second card aces may be counted as 11, thus giving some extra chances to get closer to 110 without going over, as with blackjack. Your hand is a "bust" if all your possible totals exceed 110.

In general, if you take an even number of cards, you multiply together the first two cards, the second two, etc. and then add up those products. If you take an odd number, you do the same, except that the last card's value is simply added to the sum of the products of the pairs. Let's arbitrarily agree that the dealer (computer) must hit with 75 or less, unless player has already busted, and stick at 76. I arrived at the 76 by considering that ace and six would cause a dealer to stop at blackjack, so 11 x 6 = 66 is analogous. But one would never stop there due to ability to add safely one more card, so adding a ten, we get 76. It is also nice because a player's 7-11 cards would beat a 76, just as a seven and ace would beat six and ace of dealer at a casino.

In practice, the program I've provided here does all the calculation for you, and rules are included in the program itself. You only need consider whether the next card value will be multiplied or added as part of your decision to hit or stick. Note that the program requires that you input "H" and hit <Enter> to hit, or "S" and hit <Enter> to stick.

In any case, enjoy the program. I actively solicit your comments, pertinent questions, suggestions, improvements (always plenty of room for them!) and so on. Write to me directly at the address above. For those who own TRS-80 Model 1, 3, 4 or 4P and who would prefer not to type in the program, send me a disk (or tape) and $7 (or $9.50 alone) and I'll include the program, plus an extra program or two. Please mention your computer model too. Readers with Model 3, 4 or 4P disk systems with at least one disk drive may also obtain Super-Blackjack as part of the 36 program, double volume collection available on Magic Math Plus.

Magic Math comes on a self-booting disk with XDOS, a licensed disk operating system compatible with TRSDOS 1.3, from Mr. Jim Kyle of the Software Factory. The collection is totally menu generated (again, courtesy of software from the Software Factory and available from Recreational Mathemagical Software). This combination makes use of Magic Math almost automatic.

Volume 1 contains approximately 20 programs, including Fastloan, Compound Interest, Super-blackjack, Super-Trick, Super-Fast Prime Number Cruncher, The Game of N, Fibonacci Numbers, Additive Sequences, Base Two (a trick with explanation) and loads of other goodies!

Volume 1 sells for $24.95 plus $1 for shipping and handling. A volume 2, written by programmer/writer Mr. David B. Lewis (who writes in the likes of 80 Micro and others), is available with lots of other numerical curiosities involving convergence, as in the Collatz conjecture, and with graphic and other games (as in Hexapawn, Repeater, etc.), with at least 15 programs available for the same price. Same menu generated format available for Volume 2.

Special offer: Volumes 1 and 2 combined, with the self-booting disk, XDOS, a minimum of 36 programs (including some bonuses and tutorials) in a menu generated format, a title page and four menu programs / pages - all for $36 (that's just $1 per program) plus $1 shipping and handling.

Lastly: I will be pleased to send anybody who wishes further information a catalog if you send a self-addressed stamped envelope. I remind you again that your comments and questions are welcome and appreciated. Until next issue: Happy Computing!

Dr. Michael W. Ecker
Recreational Mathemagical Software
129 Carol Drive
Clarks Summit, Pennsylvania 18411
(717) 586-2784

```
1 REM SUPERBLACK, by Dr. Michael W. Ecker, copyright 1984
6 CLS:PRINT @273, CHR$(23) "SUPER-BLACKJACK"
7 FOR DELAY=1 TO 1000:NEXT:PRINT:PRINT "        The game of
110":FOR DELAY=1 TO 2000:NEXT
8 PRINT CHR$(28):PRINT: PRINT "          A bonus from
Recreational Mathemagical Software"
9 PRINT: PRINT "          copyright 1984, Dr. Michael W. Ecker
":FOR DELAY=1 TO 2500:NEXT
10 CLS:PRINT:INPUT "WOULD YOU LIKE INSTRUCTIONS (Y or
N)";Q$
11 IF Q$="Y" THEN GOSUB 1000: REM INSTRUCTIONS
15 CLS:DEFINT A-Z
16 INPUT "FOR INCREASED RANDOMNESS, PLEASE INPUT A 2 OR
3 DIGIT NUMBER"; R
17 CLS: FOR A=1 TO R: R=RND(13): NEXT A
20 DIM PLA(20,2), DEAL(20,2): REM OUGHT TO COVER ENOUGH
CARDS & FIRST ACES
30 DIM PS(20,2),DS(20,2): REM PLAYER AND DEALER SCORES
40 PCARD=PCARD+1:DCARD=DCARD+1: REM CARDS FOR PLAYER
AND DEALER
50 PLA(PCARD,1)=RND(13): IF PLA(PCARD,1)=0 THEN 50
60 DEAL(DCARD,1)=RND(13): IF DEAL(DCARD,1)=0 THEN 60
70 IF PCARD<3 AND PLA(PCARD,1)=1 THEN PLA(PCARD,2)=11:
REM USE  ACE AS 1 OR 11
80 IF DCARD<3 AND DEAL(DCARD,1)=1 THEN DEAL(DCARD,2)=11:
REM 1st 2 CARDS ONLY
90 GOSUB 410: REM LOOK FOR PICTURE CARD EQUIVALENTS (11,
12, 13)
100 IF PCARD<2 THEN 40
110 PRINT "PLAYER'S FIRST TWO CARDS' VALUES ARE:
";PLA(1,1);PLA(2,1): PRINT
120 PRINT "DEALER'S FIRST TWO CARDS' VALUES ARE:
";DEAL(1,1);" HIDDEN": PRINT
```

```
130 PRINT: PRINT "POSSIBLE PLAYER SCORES: "
140 PI(1)=PLA(1,1)*PLA(2,1): REM PRODUCT OF FIRST 2 CARDS
FOR PLAYER
150 DI(1)=DEAL(1,1)*DEAL(2,1): REM DITTO FOR DEALER
160 GOSUB 600: REM COMPUTE POSSIBLE PLAYER SCORES
180 GOSUB 710: REM COMPUTE POSSIBLE DEALER SCORES
190 GOSUB 460: REM PLAYER TO DRAW CARDS? (HIT OR STICK?)
 00 PRINT: PRINT "PLAYER, YOUR BEST SCORE IS ";:PMAX=0
210 IF PI(2)=0 THEN PMAX=PS(PCARD,1)
220 IF PMAX>110 THEN PMAX=0: GOTO 260
230 IF PS(PCARD,1)<=110 THEN PMAX=PS(PCARD,1) ELSE PMAX=0
240 IF PS(PCARD,2)<=110 AND PS(PCARD,2)>PMAX THEN
PMAX=PS(PCARD,2)
250 PRINT PMAX: PRINT : IF PCARD>=3 AND ST$<>"S" AND
PMAX>0 THEN GOSUB 460: GOTO 200
260 IF PMAX=0 THEN PRINT "... YOU LOSE!
SORRY!":DWIN=DWIN+1: GOTO 870
270 FOR DELAY=1 TO 1000: NEXT : PRINT "LET'S SEE WHAT I
HAVE..."
280 INPUT "HIT RETURN TO CONTINUE";Q$: CLS
285 PRINT "THE DEALER'S CARDS ARE: ";DEAL(1,1);DEAL(2,1)
290 PRINT: PRINT "THE DEALER'S BEST SCORE SO FAR
IS";:DMAX=0
300 IF DI(2)=0 THEN DMAX=DS(DCARD,1)
310 IF DMAX>110 THEN DMAX=0: GOTO 350
320 IF DS(DCARD,1)<=110 THEN DMAX=DS(DCARD,1) ELSE DMAX=0
330 IF DS(DCARD,2)<=110 AND DS(DCARD,2)>DMAX THEN
DMAX=DS(DCARD,2)
340 PRINT DMAX: INPUT "HIT RETURN TO CONTINUE...";Q$
350 IF DMAX=0 THEN PWIN=PWIN+1: PRINT: PRINT "... YOU
WIN!": GOTO 870
360 PRINT : IF DMAX>75 THEN PRINT "DEALER MUST STICK.":
ELSE 820
370 IF DMAX>PMAX THEN DWIN=DWIN+1: PRINT "I WIN!": GOTO
870
380 IF PMAX>DMAX THEN PWIN=PWIN+1: PRINT "YOU WIN!":
GOTO 870
390 IF DMAX=PMAX THEN PRINT "A TIE!": GOTO 870
400 END
410 FOR PICTURE=11 TO 13
 20 IF PLA(PCARD,1)=PICTURE THEN PLA(PCARD,1)=10
430 IF DEAL(DCARD,1)=PICTURE THEN DEAL(DCARD,1)=10
440 NEXT PICTURE
450 RETURN
460 PRINT: PRINT : PRINT "PLAYER, DO YOU WANT TO HIT OR
STICK?"
470 PRINT "TO STICK, TYPE 'S' AND HIT THE RETURN KEY."
480 PRINT "TO HIT, TYPE 'H' AND HIT THE RETURN KEY."
490 ST$=""
500 INPUT ST$
510 IF ST$<>"S" AND ST$<>"H" THEN 470
520 IF ST$="S" THEN RETURN : ELSE PCARD=PCARD+1
530 PLA(PCARD,1)=RND(13): IF PLA(PCARD,1)=0 THEN 530
540 GOSUB 410
550 PRINT : PRINT "PLAYER, YOUR CARDS SO FAR ARE:"
560 FOR K=1 TO PCARD
570 PRINT PLA(K,1);
580 NEXT K
590 PRINT : PRINT "YOUR POSSIBLE SCORES ARE:"
600 PKOUNT=1: REM COUNT POTENTIAL NUMBER OF DIFFERENT
PLAYER SCORES
610 PS(PCARD,PKOUNT)=PI(PKOUNT)
620 FOR PAIR=4 TO PCARD STEP 2
630 PS(PCARD,PKOUNT)=PS(PCARD,PKOUNT)+PLA(PAIR-
1,1)*PLA(PAIR,1)
640 NEXT PAIR
650 IF PCARD/2>INT(PCARD/2) THEN
PS(PCARD,PKOUNT)=PS(PCARD,PKOUNT)+PLA(PCARD,1)
660 IF PS(PCARD,PKOUNT)<=110 AND PS(PCARD,PKOUNT)>0 THEN
PRINT PS(PCARD,PKOUNT);: ELSE IF PS(PCARD,PKOUNT)>0 THEN
PRINT "BUST ";
670 IF PKOUNT=2 THEN RETURN
680 IF PLA(1,1)=1 OR PLA(2,1)=1 THEN PKOUNT=PKOUNT+1 ELSE
RETURN
690 IF PLA(1,1)=1 THEN PI(2)=PLA(1,2)*PLA(2,1) ELSE
PI(2)=PLA(1,1)*PLA(2,2)
700 GOTO 610
710 DKOUNT=1: REM COUNT POTENTIAL NUMBER OF DEALER
SCORES
720 DS(DCARD,DKOUNT)=DI(DKOUNT)
730 FOR PAIR=4 TO DCARD STEP 2
740 DS(DCARD,DKOUNT)=DS(DCARD,DKOUNT)+DEAL(PAIR-
1,1)*DEAL(PAIR,1)
750 NEXT PAIR
760 IF DCARD/2>INT(DCARD/2) THEN
DS(DCARD,DKOUNT)=DS(DCARD,DKOUNT)+DEAL(DCARD,1)
770 IF DS(DCARD,DKOUNT)<=110 AND DS(DCARD,DKOUNT)>0 AND
DCARD>2 THEN PRINT DS(DCARD,DKOUNT);: ELSE IF
DS(DCARD,DKOUNT)>0 AND DCARD>2 THEN PRINT "BUST ";
780 IF DKOUNT=2 THEN RETURN
790 IF DEAL(1,1)=1 OR DEAL(2,1)=1 THEN DKOUNT=DKOUNT+1
ELSE RETURN
800 IF DEAL(1,1)=1 THEN DI(2)=DEAL(1,2)*DEAL(2,1) ELSE
DI(2)=DEAL(1,1)*DEAL(2,2)
810 GOTO 720
820 DCARD=DCARD+1
830 DEAL(DCARD,1)=RND(13): IF DEAL(DCARD,1)=0 THEN 830
840 GOSUB 410: PRINT "DEALER'S CARDS ARE:"
850 FOR K=1 TO DCARD: PRINT DEAL(K,1);: NEXT K
860 PRINT: PRINT "DEALER'S POSSIBLE SCORES ARE:": GOSUB
710: GOTO 290
870 PRINT : PRINT "THE TALLY: YOU'VE WON"PWIN"GAMES AND
I'VE WON"DWIN
880 INPUT "TO PLAY AGAIN, HIT RETURN. TO RETURN TO BASIC,
TYPE B FIRST."; Q$
885 IF Q$="B" THEN END
890 FOR J=1 TO DCARD:DEAL(J,1)=0:DEAL(J,2)=0: NEXT
900 FOR J=1 TO PCARD:PLA(J,1)=0:PLA(J,2)=0: NEXT
910 FOR J=1 TO DCARD: FOR K=1 TO DKOUNT:DS(J,K)=0: NEXT :
NEXT
920 FOR J=1 TO PCARD: FOR K=1 TO PKOUNT:PS(J,K)=0: NEXT :
NEXT
930 PCARD=0:DCARD=0: CLS : GOTO 40
940 END
1000 CLS: PRINT "SUPER-BLACKJACK IS PLAYED A LOT LIKE
BLACKJACK."
1010 PRINT: PRINT "YOU ARE DEALT TWO CARDS, AS IS THE
DEALER (THE COMPUTER)."
1020 PRINT "INSTEAD OF ADDING THE CARDS' FACE VALUES,
THEY ARE MULTIPLIED."
1030 PRINT "PICTURE CARDS ARE AUTOMATICALLY REPORTED
AS HAVING A VALUE OF"
1040 PRINT "10. AN ACE COUNTS AS 1, EXCEPT THAT ON THE
FIRST 2 CARDS, YOU"
1050 PRINT "MAY ALSO COUNT AN ACE AS AN 11 IF YOU WISH.
THE COMPUTER WILL"
1060 PRINT "HANDLE THAT FOR YOU AUTOMATICALLY,
HOWEVER."
1065 PRINT:PRINT "THE OBJECT IS TO AMASS A TOTAL CLOSER
TO 110 THAN THE DEALER"
1066 PRINT "WITHOUT GOING OVER 110."
1070 PRINT:PRINT: INPUT "Hit <ENTER> to continue";X
1080 CLS: PRINT "NOW, THIS GAME IS ACTUALLY A BIT MORE
INVOLVED, AND A BETTER"
1090 PRINT "NAME WOULD BE THE GAME OF 110. YOU SEE, 10 X
11 = 110. DO YOU"
1100 PRINT "SEE THE ANALOGY TO BLACKJACK?"
1110 PRINT:PRINT "NOW HERE IS THE CATCH. FIRST OF ALL,
BEYOND THE 2 CARDS, IF"
1120 PRINT "YOU CHOOSE TO TAKE A THIRD CARD, THAT VALUE
IS NOW ADDED TO"
1130 PRINT "THE PREVIOUS PRODUCT. FOR INSTANCE, SAY YOU
HAVE THE CARDS"
1140 PRINT "10 AND 7. IF YOU STICK (STAND), THEN YOUR
SCORE IS 70, WHICH"
1150 PRINT "IS ONLY A FAIR SCORE. BUT IF YOU TAKE A HIT
(ANOTHER CARD),"
1160 PRINT "THEN YOU CAN GET CLOSER. SAY YOU DRAW A 6.
NOW THAT 6 IS ADDED"
1170 PRINT "TO YOUR PREVIOUS 70 TO GIVE 76, A BIT BETTER."
1180 PRINT:PRINT: INPUT "Hit <ENTER> to continue"; X
1190 CLS:PRINT "IF YOU TAKE ANOTHER CARD NOW, YOU HAVE
TO RE-CALCULATE THE"
1200 PRINT "SCORE. (Of course, the computer does it for you.)
NAMELY,"
1210 PRINT "TAKE THE LAST 2 SCORES AND MULTIPLY THEM
TOGETHER. SAY YOU"
1220 PRINT "NOW GET A 3. YOUR CARDS ARE 10, 7, 6, 3. YOU
MULTIPLY 10 BY 7"
1230 PRINT "AND 6 BY 3, THEN ADD THE PRODUCTS. YOU GET 70
+ 18 = 88."
1240 PRINT:PRINT "IF YOUR 4th CARD WERE A 10 INSTEAD OF A
```

```
3, YOUR TOTAL WOULD"
1250 PRINT "BE OVER 110 AND YOUR HAND WOULD 'BUST' (OR
'BREAK'), AND YOU"
1260 PRINT "WOULD THEN BE THE LOSER OF THAT HAND.
SUPPOSE, THOUGH, THAT YOU"
1270 PRINT "HAVE YOUR 10, 7, 6, 3 AGAIN, FOR A SCORE OF 88.
IF YOU TAKE A"
1280 PRINT "5th CARD, THEN IT IS ADDED TO 88."
1290 PRINT: PRINT: INPUT "Hit <ENTER> to continue"; X
1300 CLS: PRINT "IN OTHER WORDS, THE SCORING IS: IF YOU
HAVE AN EVEN NUMBER"
1310 PRINT "OF CARDS, THE FIRST TWO ARE MULTIPLIED, THEN
THE SECOND TWO,"
1320 PRINT "AND SO ON. THE PRODUCTS ARE ADDED TOGETHER.
FOR AN ODD NUMBER"
1330 PRINT "OF CARDS, DO THE SAME PROCESS, LEAVING THE
LAST CARD. THEN JUST"
1340 PRINT "ADD THAT VALUE TO THE RESULT."
1350 PRINT: PRINT "AS WITH 21, CARDS ARE DEALT FACE UP,
EXCEPT FOR DEALER'S 2nd."
1360 PRINT "THAT CARD REMAINS HIDDEN UNTIL AFTER YOU,
THE PLAYER, HAVE"
1370 PRINT "DRAWN AS MANY CARDS AS YOU WISH. THEN
YOU'LL SEE DEALER'S HAND."
1375 PRINT: INPUT "Hit <ENTER> to continue";X
1380 CLS:PRINT:PRINT "NOTE: THERE IS NO BETTING,
INSURANCE, SPLITTING OR DOUBLING."
1390 PRINT "HOWEVER, THE COMPUTER DOES KEEP TRACK OF
THE NUMBER OF GAMES"
1400 PRINT "WON BY THE PLAYER AND THE DEALER
(COMPUTER)."
1410 PRINT:PRINT "ONE OTHER DIFFERENCE IS THAT 110 ON
THE FIRST TWO CARDS IS NOT"
1420 PRINT "AN AUTOMATIC WINNER. THUS, IT IS POSSIBLE FOR
YOU TO CATCH UP"
1430 PRINT "TO YOUR OPPONENT IF HE GETS A NATURAL 110 BY
AN ACE AND 10."
1440 PRINT "NOTE THAT DEALER MUST HIT ON 75 OR LESS AND
STICK ON 76 OR MORE."
1450 PRINT: INPUT "TO BEGIN PLAY, HIT <ENTER>.";X
1460 CLS: PRINT @220, "GOOD LUCK!":FOR Z=1 TO 1000:NEXT
1500 RETURN
```

AUDIO AMPLIFIER IN YOUR VIDEO

by Gary Bryce

[Reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, Australia.]

Some of you may remember one club meeting a few months ago when after a long day demonstrating a particular music program, th jury rigged amplifier that I had connected to my system decided to give up the ghost in a rather spectacular manner (and boy didn't it STINK!).

This prompted me to develop a simple amplifier that gave a better quality output than the converted radios and such that we have all been using (this includes the inbuilt amplifier of the SCRAP-80 (Blue Label).

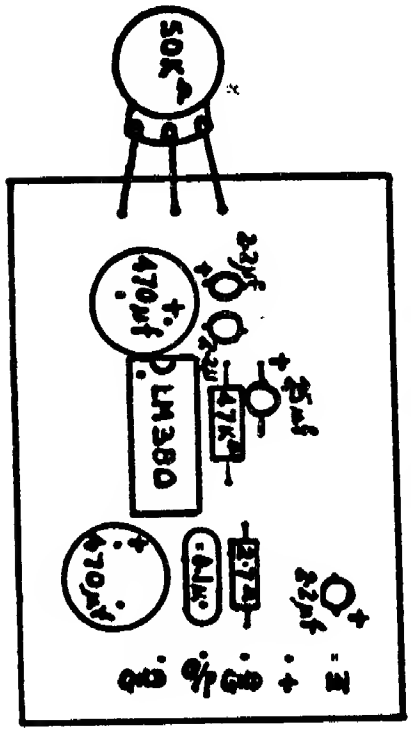
So here is a simple amp which I feel meets all the requirements I needed, I hope that it will meet your own needs. The details of the amplifier, with volume control, which mounts in and is powered from the video, follows.

PARTS REQUIRED

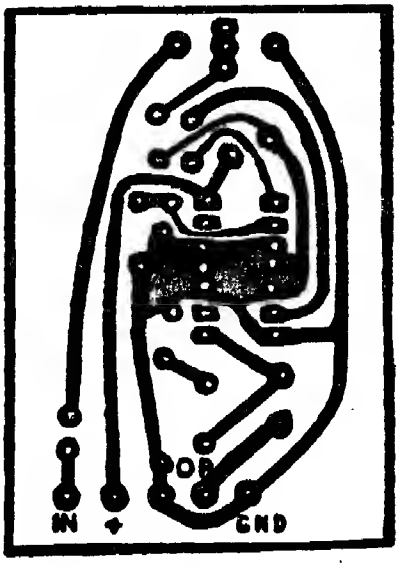
1 x LM380 Audio Power Amplifier	Z-6080
2 x 470uF Electrolytic Capacitor (PCB)	R-4420
1 x 25uF Electrolytic Capacitor (PCB)	R-4320
3 x 2.2uF Tag Tantalum Capacitor	R-4730
1 x 0.1uF Ceramic Capacitor	R-2360
1 x 2.7ohm 1/4W Resistor	R-1012
1 x 47Kohm 1/4W Resistor	R-1114
1 x 50Kohm Potentiometer (Log Scale)	R-6823

The Circuit Diagram, PCB layout and Component position drawing appear below. The Video Monitor may not be fitted with an in-built speaker, if not an 8 ohm speaker of not less than 2 Watt power rating should be used. I have left details of fitting and positioning to you, as this varies between different monitors.

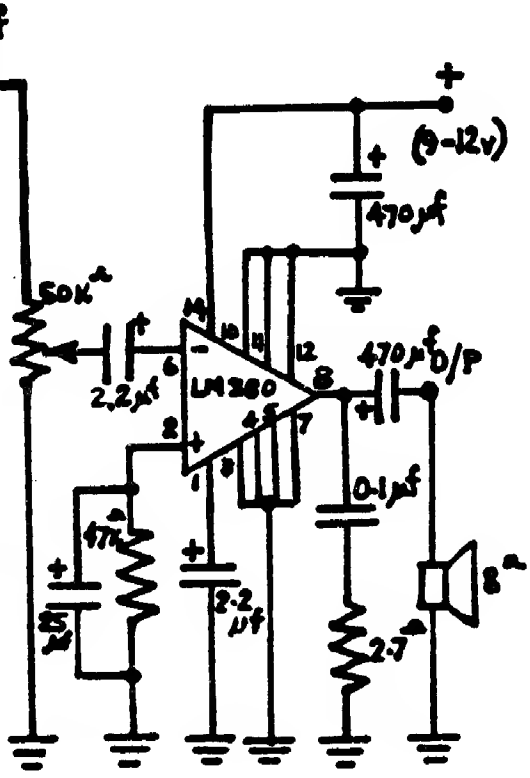
NOTE: PC BOARD LAYOUTS NOT EXACTLY TO SCALE



Top Layout



PCB Layout



Circuit

NEWDOS/80 USERS' GROUP
by Greg Small

Welcome to this new column devoted to the understanding, use and modification of Newdos/80.

First, some background on the reasons for the existence of this column.

This past summer I bought a 21 meg hard drive and Newdos/80 version 2.5 from Charley Butler at TAS. To try and eliminate the border crossing hassles from the U.S.A. to Canada I drove to Lansing to pick up my new "toys".

Charley and I spent some time reminiscing about the "good ole days" and the so-called "demise" of Newdos/80. We talked about my plans for an International Newdos/80 Users' Group as outlined in previous issues of Northern Bytes and also about the changes that many individuals have made to the DOS on their own.

We discussed the duplication of efforts that had to be occurring.

The Newdos/80 Users' Group does exist -- but for the time being only as an adjunct to these pages. My thanks to the 15 or so people who took the time to write or call. Your thoughts and ideas are appreciated.

The problem that occurred was that the people who did write, for the most part, did not live in North America - and any individual communications or information sharing were taking considerably longer than desirable to arrive at the destination. By using this medium we will be able to pass along information on a more timely basis.

My idea for this column is to have a central storehouse of ZAPs, ideas, documentation, etc. that can be drawn from as needed. Some of you wanted technical topics discussed. Others said they would be satisfied if the darn manual was explained in "real English sentences". Others wanted to know specific shortcuts that could be used. I hope to address portions of all of those things in these pages and will appreciate any specific comments, requests or criticisms.

Some of the other things that I want to detail here are user generated ZAPs to either the DOS or application programmes. Besides documenting the patches I will also discuss the various means of developing these patches.

I want to cover the differences between the Model I and III sions and the floppy and hard drive versions.

If you have any topics you want discussed or specific questions you would like to have answered feel free to voice them.

[I also hope to convince Jack to add public domain disks specifically devoted to ideas and files from these columns and specific Newdos/80 utilities.]

Next, a little background about me. I bought a 16K Model I tape system in December 1979. This was a hobby that could pay for itself -- or so I told my wife.

Well, currently I own 2 Model I's, two 4's, a 100, and a PC-1. I also have nine floppies, a 21 meg hard drive, two modems, three printers, two printer buffers and lots of cooling fans. And it now does pay for itself. [All this equipment also does a good job of heating the basement office even when the outside temperature reaches -20 degrees F.]

I am using a self-written BASIC program to maintain a 20,000 name mailing list for a local charitable organization. Without Newdos/80's "funny files" this would not have been practical.

But -- and this is critical -- I learned most EVERYTHING I know about computers from reading 80-Micro, The Alternate Source Programmer's Journal, 80-U.S., Dennis Kitsz' long gone newsletters, Bill Barden's Disk Interfacing Guide, much of the IJG series, Hubert Howe's early columns in Computronics Newsletter and, of course, Jack Decker's TRS-80 Rom Routines Documented ($19.95 plus $3.00 shipping from The Alternate Source).

The other major learning tools I have used are Newdos/80's disassembler, TASMON and any machine language program I got my hands on. Commenting a disassembly and patching a program is invaluable experience in learning machine code.

Which brings us to this issue's primary topic, Newdos/80's BOOT/SYS commented.

I will assume that all of you who care, know how the TRS-80 boot routine gets BOOT/SYS into memory. Once it is in memory the HL register is set to 4200H (4300H in the III/4) and control is transferred to the program (BOOT/SYS in this case) at that address.

Following is the commented code I put together for the Model I Version. I have much of the Model III Version commented and will include that in a future column.

Newdos/80 Version 2.0 (Model I) - single density - commented disassembly of BOOT/SYS. The additional lines that start with ";*" are for the double density version with a single density track 0.


```
;-------------------------------------------------------------------------
; SETUP - get ready to read SYS0/SYS from the disk
;-------------------------------------------------------------------------
 4200 00      NOP             ;Do nothing
 4201 FE11    CP    11H       ;4202 = Directory "track"
                              ;This is another do nothing instruction
 4203 F3      DI              ;Turn off the interrupts
 4204 21EC37  LD    HL,37ECH  ;Point to the command/status register
 4207 36FE    LD    (HL),0FEH ;Ensure we are set to SINGLE density
;*4207 36FE    LD    (HL),0FFH ;Ensure we are set to DOUBLE density
 4209 36D0    LD    (HL),0D0H ;Force an interrupt on the FDC
 420B 23      INC   HL        ;Point to the track update register
 420C 3600    LD    (HL),00H  ;Store a 0 in it for track zero
 420E 23      INC   HL        ;Increment to point to sector register
 420F 3600    LD    (HL),00H  ;Store a zero in it for sector zero
 4211 110500  LD    DE,0005H  ;Set DE for track 0 sector 5
;*4211 110501  LD    DE,0105H  ;Set DE for track 1 sector 5
 4214 D9      EXX             ;Store it for disk read
 4215 31E041  LD    SP,41E0H  ;Set up a new stack area out of the way
 4218 21FF51  LD    HL,51FFH  ;Point to the end of temporary buffer
;-------------------------------------------------------------------------
; MOVE LOADER - processes loader codes
;-------------------------------------------------------------------------
 421B CD5242  CALL  4252H     ;Call MOVEDATA IN subroutine
 421E FE20    CP    20H       ;Is value returned valid loader code
 4220 47      LD    B,A       ;Save code in B for later test of type
 4221 3029    JR    NC,424CH  ;If not valid code; go to NOSYS routine
 4223 57      LD    D,A       ;Put the loader code in D
 4224 CD5242  CALL  4252H     ;Call MOVEDATA IN subroutine
 4227 4F      LD    C,A       ;Put the quantity to load in C
 4228 CD5242  CALL  4252H     ;Call MOVEDATA IN sub
 422B 5F      LD    E,A       ;Put LSB of load address in E
 422C 1012    DJNZ  4240H     ;DEC loader code, if NZ then check for
                              ;transfer at LOADER >1 subroutine
 422E CD5242  CALL  4252H     ;Call MOVEDATA IN sub
 4231 57      LD    D,A       ;Put MSB of load address in D
                              ; thus making DE point to load address
 4232 0D      DEC   C         ;Adjust number of bytes to load
 4233 0D      DEC   C         ; twice
;-------------------------------------------------------------------------
; GET MORE - bytes to put in place
;-------------------------------------------------------------------------
 4234 2C      INC   L         ;Increment pointer for temporary buffer
 4235 CC5542  CALL  Z,4255H   ;If its full, then call
                              ;MOVE DATA FROM DISK subroutine
 4238 7E      LD    A,(HL)    ;Else put the next byte in A
 4239 12      LD    (DE),A    ;and move it to the proper place
 423A 13      INC   DE        ;Increment the proper place
;-------------------------------------------------------------------------
; THROWAWAY - the comments
;-------------------------------------------------------------------------
 423B 0D      DEC   C         ;Reduce the number of bytes to move
 423C 20F6    JR    NZ,4234H  ;If its not zero go to GET MORE
 423E 18DB    JR    421BH     ;Done with comments - go to MOVE LOADER
;-------------------------------------------------------------------------
; LOADER >1 - checks for comments
;-------------------------------------------------------------------------
 4240 10F9    DJNZ  423BH     ;If loader >01H then go to THROWAWAY
 4242 CD5242  CALL  4252H     ;Call MOVEDATA IN
 4245 57      LD    D,A       ;Get MSB of transfer address into D
 4246 1A      LD    A,(DE)    ;Get first byte of SYS0/SYS in A
 4247 FEA5    CP    0A5H      ;Is it SYSTEM CHECK byte (A5H)
 4249 13      INC   DE        ;Move to next byte
 424A D5      PUSH  DE        ;Put it on the stack
 424B C8      RET   Z         ;If it was 0A5H "RET" to it, jumping
                              ;into SYS0/SYS at 4D01H
 424C 21E542  LD    HL,42E5H  ;Else point to NOSYS message
 424F C3C342  JP    42C3H     ;Display it
;-------------------------------------------------------------------------
; MOVEDATA IN - from buffer or from disk if buffer empty
;-------------------------------------------------------------------------
 4252 2C      INC   L         ;Move temp buffer pointer - if increment
                              ;from FFH to 00H then Z flag is set
 4253 7E      LD    A,(HL)    ;Get byte from HL in A
```

```
4254 C0      RET   NZ          ;Return if Zero flag is off - fall
                               ;through if past end of buffer
;-----------------------------------------------------------------
4255 D9      EXX               ;Restore registers for reading disk
4256 060A    LD    B,0AH       ;Store number of times to try (10) in B
;-----------------------------------------------------------------
; READ DISK
;-----------------------------------------------------------------
4258 21E137  LD    HL,37E1H    ;Point to drive select register
425B 3601    LD    (HL),01H    ;Select drive zero
425D D5      PUSH  DE          ;Save DE - track and sector numbers
425E C5      PUSH  BC          ;And BC - only need B for # of retries
425F 7B      LD    A,E         ;Load A from E - sector number
4260 D60A    SUB   0AH         ;Subtract # of sectors on front of disk
;*4260 D612  SUB   12H         ;Subtract # of sectors on front of disk
4262 3803    JR    C,4267H     ;If < test value JR to CONTINUE
;-----------------------------------------------------------------
; CHANGE - sides
;-----------------------------------------------------------------
4264 5F      LD    E,A         ;Get result in E - will be sector number
                               ;on back side
4265 3609    LD    (HL),09H    ;Store 0000 1001 in drive select reg
                               ;This selects drive 0 and switches to
                               ;back side
;-----------------------------------------------------------------
; CONTINUE - don't change sides
;-----------------------------------------------------------------
4267 21EC37  LD    HL,37ECH    ;Point to Command/status register
426A CDCE42  CALL  42CEH       ;Call CHECK STATUS
426D ED53EE37 LD   (37EEH),DE  ;Put track and sector no. in register
4271 361B    LD    (HL),1BH    ;SEEK at 40ms and verify head position
4273 CDCE42  CALL  42CEH       ;Call CHECK STATUS
4276 3688    LD    (HL),88H    ;Read 1 IBM sector w/o head settle delay
4278 11EF37  LD    DE,37EFH    ;Point to data register
427B 010051  LD    BC,5100H    ; and temporary buffer
427E CDD742  CALL  42D7H       ;Call DELAY routine
4281 7E      LD    A,(HL)      ;Get status from CMD/STATUS register
4282 E683    AND   83H         ;Test it for READY
4284 E28142  JP    PO,4281H    ;Loop till READY
;-----------------------------------------------------------------
; STORE BYTE - in buffer
;-----------------------------------------------------------------
4287 1A      LD    A,(DE)      ;Load A from the data register
4288 02      LD    (BC),A      ;Put it where BC points in temp. buffer
4289 03      INC   BC          ;Bump BC pointer one higher
;-----------------------------------------------------------------
; START LOOKING - for a byte to be READY
;-----------------------------------------------------------------
428A CB4E    BIT   01H,(HL)    ;Test for DRQ
428C C28742  JP    NZ,4287H    ;Yes - loop to STORE BYTE
428F CB4E    BIT   01H,(HL)    ;DRQ now?
4291 C28742  JP    NZ,4287H    ;Yes - loop to STORE BYTE
4294 CB4E    BIT   01H,(HL)    ;DRQ yet?
4296 20EF    JR    NZ,4287H    ;Yes - loop to STORE BYTE
4298 CB46    BIT   00H,(HL)    ;Test for busy
429A 2808    JR    Z,42A4H     ;No - go to LOOK AGAIN
429C CB4E    BIT   01H,(HL)    ;DRQ yet?
429E 20E7    JR    NZ,4287H    ;Yes - loop to STORE BYTE
42A0 CB7E    BIT   07H,(HL)    ;Test for Not READY - to
                               ;eliminate possibility of silent death
42A2 28E6    JR    Z,428AH     ;If still READY go to START LOOKING
;-----------------------------------------------------------------
; LOOK AGAIN - for a byte to be READY
;-----------------------------------------------------------------
42A4 7E      LD    A,(HL)      ;Get status in A
42A5 36D0    LD    (HL),0D0H   ;Force an interrupt
42A7 C1      POP   BC          ;Restore BC for times to test
42A8 D1      POP   DE          ;and DE for track and sector numbers
42A9 E6FC    AND   0FCH        ;Test for DRQ and/or BUSY
42AB 200C    JR    NZ,42B9H    ;Neither - goto TRY AGAIN
42AD 1C      INC   E           ;Bump sector pointer
42AE 7B      LD    A,E         ;Put it in A
42AF D60A    SUB   0AH         ;Test for > # of sectors per track
;*42AF D612  SUB   12H         ;Test for > # of sectors per track
42B1 2003    JR    NZ,42B6H    ;Not >, then skip SECTOR INCREMENT
;-----------------------------------------------------------------
; SECTOR INCREMENT - at track boundary
;-----------------------------------------------------------------
42B3 14      INC   D           ;Increment track number
42B4 1E00    LD    E,00H       ;Set sector number to 0
;-----------------------------------------------------------------
; SAME TRACK - so don't increment
;-----------------------------------------------------------------
42B6 D9      EXX               ;Exchange registers
42B7 7E      LD    A,(HL)      ;Load first data byte into A
42B8 C9      RET   .           ;Return to caller
;-----------------------------------------------------------------
; TRY AGAIN
;-----------------------------------------------------------------
42B9 CDD742  CALL  42D7H       ;Call WASTE TIME
42BC 360B    LD    (HL),0BH    ;Restore head
42BE 1098    DJNZ  4258H       ;Not 10 tries - goto READ DISK
42C0 21DD42  LD    HL,42DDH    ;Point to ERROR message string
;-----------------------------------------------------------------
; SHOW IT - one byte at a time
;-----------------------------------------------------------------
42C3 7E      LD    A,(HL)      ;Put the byte in A
42C4 FE03    CP    03H         ;Is it a terminator
42C6 28FB    JR    Z,42C3H     ;YES - then infinite loop
42C8 23      INC   HL          ;NO - then bump pointer to next char
42C9 CD3300  CALL  0033H       ;Call single byte display ROM routine
42CC 18F5    JR    42C3H       ;Go back to SHOW IT again
;-----------------------------------------------------------------
; TEST BUSY and advise status
;-----------------------------------------------------------------
42CE CDD742  CALL  42D7H       ;Call DELAY routine
42D1 CB46    BIT   00H,(HL)    ;Test bit 0 of HL for BUSY
42D3 20FC    JR    NZ,42D1H    ;Loop back to TEST till not BUSY
42D5 7E      LD    A,(HL)      ;Get status into A
42D6 C9      RET               ;Back to caller
;-----------------------------------------------------------------
; DELAY
;-----------------------------------------------------------------
42D7 3E06    LD    A,06H       ;Store delay value in A
42D9 3D      DEC   A           ;Decrement by one
42DA 20FD    JR    NZ,42D9H    ;Not 0 yet - back to decrement it again
42DC C9      RET               ;Done - so go back to caller
;-----------------------------------------------------------------
; ERROR - message
;-----------------------------------------------------------------
42DD 1C      DEFB  1CH         ;Code for HOME CURSOR
42DE 1F      DEFB  1FH         ;Code for ERASE TO END OF SCREEN
42DF 45      DEFM  'ERROR'     ;ERROR message string
....
42E4 03      DEFB  03H         ;Terminator
;-----------------------------------------------------------------
; NO SYS - message
;-----------------------------------------------------------------
1 42E5 1C    DEFB  1CH         ;Code for HOME CURSOR
42E6 1F      DEFB  1FH         ;Code for ERASE TO END OF SCREEN
42E7 4E      DEFM  'NO SYS'    ;NO SYS message string
....
42ED 03      DEFB  03H         ;Terminator
;-----------------------------------------------------------------
42EE 00      NOP               ;The 15 bytes from 42EEH through 42FCH
....                           ;are not used and could be used for
42FC 00      NOP               ;disk identification or other things
;-----------------------------------------------------------------
42FD 00      DEFB  00H         ;
;*42FD 00    DEFB  00H         ;The 3 bytes at 42FDH through 42FFH are
42FE 07      DEFB  07H         ;moved to the SYSFLAG table indexed
;*42FE 50    DEFB  50H         ;from 4380H.  They are moved at 402BH
42FF 00      DEFB  00H         ;and 4035H of SYS0/SYS
;*42FF 03    DEFB  03H         ;
```

Well, that is fine. But what REAL good is it? What can be done with it?

Here are some suggestions and things for you to think about:

At 4257H we could change the number of retries from 0AH (10) to some other value to allow for a faster error routine with bad disks.

At 425CH and 4266H we could change the code that selects drive 0 to experiment with reading SYS0/SYS from another drive.

At 4272H we could change the SEEK rate from 40ms to 20ms or 5ms to allow faster booting. This, of course, would only work if our drives allow this.

At 42C6H we could change the jump relative (JR) that locks the system up when the NO SYS or ERROR message is displayed to jump to the 15 bytes starting at 42EEH. This could allow a JP to TASMON or another piece of code that was left in memory or could check for a key being pressed to start the boot sequence over.

The bytes at 42EEH could also be used to display verification that certain portions of BOOT/SYS have been successfully completed.

I will leave these ideas with you. If you do make some interesting modifications please write them up and send them to me for inclusion here for the rest of the readers. Don't worry about the format you use. I would rather get your changes on the back of n old envelope written in crayon than not see them at all.

Next, I want to tell you of some very interesting experiments I have made with my Model I system. I ran my BBS on a Model I with floppies for about 12 months.

After a while I began to expect to find the system hung up after an attempted reset about two or three times per month. This was the old Model I reset problem. Many of us have pressed the reset button and found that the Model I did not properly reset. So we pressed it again and thought it was not pressed firmly enough the first time.

But when you run a board 24 hours per day you begin to realize that there is a definite problem with the re-booting of a Model I.

I cleaned the connectors. When that did not solve the problem for long I installed gold plugs. That worked for a while but I was soon back to the same old problem. Next I made patches to TBBS that were suggested by the author. I also patched Newdos/80 to allow a longer head settle time on re-boot. Those things also did not solve the problem.

Next I decided to take the bull by the horns. I patched TBBS's reset code to jump to 0066H on the recommendation of a friend who made his system boot from double density and/or 8 inch drives. That did not help.

I added a FORCE INTERRUPT instruction also on his suggestion. Still no success.

So finally I changed the patch to jump to a piece of code I wrote that resided in high memory. There I duplicated the ROM boot routine. But to prove my work and further the discovery process I had this routine print a letter on the screen after each major subroutine was successfully executed. Reset was more reliable for a while -- but then it would go bad again.

However, now I had some clues as to where the problem was located. Or did I? Each time I found the system hung I also found the letters A-L on the screen and L was the letter that was printed just before the execution of the JP 4200H instruction in my relocated "ROM code".

So, possibly the problem was not the Model I but was really Newdos/80's BOOT. This was the reason for the foregoing commented disassembly - but all seemed well in the code.

My next theory was that memory was getting "farkled" in the 4200H - 42FFH page and BOOT/SYS was therefore corrupted and hanging the system. I installed code to do a checksum on that page and test for a successful load from disk. If the code passed the test then an "M" was added to the A-L already on the screen.

The next time the system hung the letter M was showing! So BOOT was successfully loaded and somewhere after the jump to 4200H the system hung.

I decided then that it was time I bought a Model 4. I put the board on it and have not had the system hang since then.

What does all this lead me to think? The Model I has a very definite problem and I have NOT found it but have eliminated lots of suspected problems. I really do not think it is Newdos/80 as Model I DOS users of all flavours complain of these problems.

Does anyone have any other thoughts?

Now, for some miscellaneous house-keeping. You may contact me in one of the following ways:

By mail: Greg Small
Box 607
Stouffville, Ontario, Canada
L0H 1L0

If you want a reply enclose a self-addressed #10 or larger envelope. Canadians should affix current Canadian postage. Americans may send 35 cents in coins or a buck or two and I'll buy them some Canadian stamps for future communications. People from other countries may send International Reply Coupons for Air Mail Postage or U.S. or Canadian currency as outlined above.

By phone: (416) 640-4400 from 7PM to 9PM most week nights and from 10AM to 6PM most weekends. If there is no answer call back later. Please note I am in the Eastern time zone.

By dataline: (416) 640-3434 - 24 hours a day. This is my BBS. It runs TBBS software at 300 baud with a protocol setting of 8N1. As there is a pre-use qualification questionnaire that must be filled out before full access can be gained I suggest you prepare your message off-line and then upload it as a Message to Sysop at the <&> prompt which will be available on your first call. Then call back in 3 - 4 days to retrieve any answers I leave.

I will be setting up a Newdos/80 User's Group section on the board that will be open to all Northern Bytes readers who seek access.

By MCI Mail: User name: INTL6
ID #: 248-6778

Note that the MCI Mail account is a temporary one and will be replaced sometime in early 1985 (if you find it doesn't work, try the user ID #: 251-7579).

That is all for this issue. There are lots of exciting plans and releases on the horizon and I will try and outline some of these next issue.

One major thing I will do next issue is show the method I used to to de elop a few ZAPs to have the DOS do what I want it to do.

These include patching the system so that every utility program shows spaces as spaces and not as periods, a drive activity indicator for the hard disk system, and ZAPs to allow use of the filespec.ext convention for users who are also saddled with CP/M or MS-DOS machines.

Please write and tell me what you are doing and/or what you want to see discussed here.

PATCHES TO CHANGE CERTAIN CHARACTERISTICS OF TASMON

by Jack Decker

These patches are optional and may be applied as desired to your working copy of TASMON. They may be applied in memory only, in which case they will remain in effect until the end of your session with TASMON, or you may use TASMON's W D command to write a copy of your patched TASMON to disk.

PLEASE NOTE that these patches apply only to the following versions of TASMON, however, information is given to allow you to find the proper location to apply the patch if you have another version.

MODEL I: ver. 2.22 MODEL III: ver. D7 MODEL 4: ver. 1.11

1. CHANGE FIRST CHARACTER OF LABELS DURING DISASSEMBLY

Using TASMON's M A (Modify memory using ASCII) command, change the following byte from the letter "Z" (5AH) to the desired first character for labels:

MODEL I: 7486H MODEL III: 748AH MODEL 4: F436H

If you have a different version of TASMON, key in F nnnn 3E 5A (where nnnn is normally 6000H in the Models I & III, and E000H in the Model 4). The displayed address PLUS ONE is the byte to change.

2. REMOVE 7-BYTE "HEADER" FROM DISASSEMBLIES TO DISK

TASMON was written to output disassembled source code that could be loaded directly into Apparat's modified version of the Radio Shack Editor-Assembler program (as found on NEWDOS/80 master disks). Most Editor-Assembler programs are able to read this source code format, but some (notably Radio Shack's Series I) will not read these source code files because of the seven-byte "header" that is placed at the beginning of these files. To prevent TASMON from writing this "header", use TASMON's M H (Modify memory) command to change two bytes starting at the addresses shown below. The bytes should be changed from 3E D3 to 18 19 in the Models I & III, and to 18 0C in the Model 4. This inserts a JR instruction that bypasses the code that writes the header to disk. Change the two bytes starting at:

MODEL I: 7312H MODEL III: 731CH MODEL 4: F2EEH

If you have a different version of TASMON, key in F nnnn 3E D3 (where nnnn is normally 6000H in the Models I & III, and E000H in the Model 4) to find the starting address to change.

3. CHANGE AMOUNT OF KEY DEBOUNCE (MODELS I & III ONLY)

This patch is especially useful when you are using a speed-up modification, or when running the Model III version of TASMON on a Model 4 using the 4 MHz clock speed. Using TASMON's M H (Modify memory) command, change the following byte from 0AH to the value that gives the desired amount of keybounce control. For example, to double the normal keybounce delay, change the byte to 14H:

MODEL I: 7A10H MODEL III: 7A53H

If you have a different version of Model I or III TASMON, key in F 6000 01 00 0A. The displayed address PLUS TWO is the byte to change.

TRACE/CMD
by Greg Small

A Utility for Newdos/80 Version 2.x

The Newdos/80 routines that Tony Domigan has outlined in this issue of Northern Bytes has finally allowed me to put my TRACE/CMD routine into the public domain.

About a year and a half ago when I first was studying Newdos/80's code there were times when I was having difficulty determining what it was actually doing and to what value some of the control bytes were set during actual program execution.

I remembered the TRACE function of TRSDOS 2.3 and wished it was available to me under Newdos/80.

So, I simply went into 2.3's code and lifted the routine to display the TRACE and wrote a short routine to link that code into Newdos/80's interrupt chain. But I was reluctant to publish the code as it was still protected by copyright.

Since that time I have refined the code several times but always used Randy Cook's code to display a byte in hex to the screen. Now, however, I can publish as the code is changed to use the routine Tony has discovered.

Tony has explained the two routines so I will simply explain what I have done.

Line 180 defines FOLLOW as the byte to follow on the screen. You will find that the following may be of interest! Model I - 4369H and Model III - 4289H. This is one of the SYSFLAG bytes. Once you have the program running entering a few DOS commands will cause this byte to change. It is impossible to follow this byte's status any other way.

Lines 220 through 290 ensure we are using Newdos/80 and that we are running it on a I or a III. (I would be interested to see it running on another machine!) If we fail this test an error is displayed at lines 650 through 850.

Note the tricky use of DEFB 0DDH at line 680. This is much used in professional programs but I have yet to see it documented. The code will assemble as:

```
5249 215352          LD      HL,MSG1
524C DD              DEFB    0DDH
524D 217752          LD      HL,MSG2
5250 C36744          JP      4467H
```

but disassembles (and is seen by the Z-80) as:

```
5249 215352          LD      HL,5253H
524C DD217752        LD      IX,5277H
5250 C36744          JP      4467H
```

so that if line 240 is executed transfer goes to 5249H where HL is set to 5253H. Register IX is then set to 5277H at 524CH and the JP 4467H is executed at 5250H. If, however line 290 is executed transfer goes to 524DH which is in the MIDDLE of the LD IX,5277H instruction and is executed as LD HL,5277H and then the JP 4467H instruction occurs.

This type of code will save many bytes in a program that does a lot of this type of message displaying and, of course, other routines than message displaying could benefit from this type of code.

The saving is calculated as plus one byte for the 0DDH and minus three bytes for the JP 4467H for a net savings of two bytes per time this is used.

Lines 300 through 400 do some housekeeping if we are found to be in a Model III at line 280. Lines 420 through 520 do the actual working program relocation. The form for this patching and relocation code is taken from Jack Decker's TRS-80 ROM Routines Documented ($19.95 plus $3.00 shipping from The Alternate Source).

Lines 540 through 630 patch the working program into the DOS interrupt chain and fix HIMEM to protect the working program.

Lines 920 and 930 are for DOS's use and can be studied from the reference given in lines 890 and 900.

Lines 950 through 1000 do the actual setup and display of the byte we are following.

Line 1040 sets up a value to add to the current stack pointer. Line 1050 does the actual addition and stores the result in HL. This value is the address stored on the stack of the actual place where the interrupted program will RETurn after the background TRACE program returns control to the interrupt service routine. This, in turn, will return control to the program that is running. This MAY simply be DOS.

For some reason unknown to me the values needed for the adder are different in the two machines. The values used are 0EH (14) for the Model I and 14H (20) for the Model III. Can anyone enlighten us for the reason for this?

Lines 1070 through 1120 move this computed address to DE and display it. Then the RETurn is executed from the display routine and control is given to the interrupt service routine and eventually to the main program that was running before the interrupt occurred. This MAY simply be DOS.

Note that the code from lines 950 through 1000 could changed to display a two byte WORD or that additional code could be added here to display additional bytes of interest.

The actual program code follows:

```
                00010 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00020 ;
                00030 ;TRACE/ASM - by Greg Small - 17jan85 - 19:00
                00040 ;Installs TRACE function plus for Newdos/80 Version 2.x
                00050 ;
                00060 ;Copyright 1985 by Greg Small
                00070 ;
                00080 ;Permission to use for non-commercial applications is
                00090 ;hereby granted -- all other rights reserved.
                00100 ;
                00110 ;program from an original idea by Randy Cook
                00120 ;relocation section from an idea by Jack Decker
                00130 ;
                00140 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00150 ;
5200            00160         ORG     5200H
                00170 ;
0000            00180 FOLLOW  EQU     0000H           ;Byte to follow
                00190 ;
5200            00200 INIT    EQU     $
                00210 ;
5200 3A2744     00220         LD      A,(4427H)       ;Get Newdos/80 ID byte
5203 FE82       00230         CP      82H             ;Is it Newdos/80 Vers. 2
5205 2042       00240         JR      NZ,ERROR1       ;NO - go process error
5207 3A2B44     00250         LD      A,(442BH)       ;Get Model number
520A FE01       00260         CP      01H             ;Is it a I
520C 2824       00270         JR      Z,RELOC         ;YES - then go to RELOC
520E FE03       00280         CP      03H             ;Is it a III
5210 203B       00290         JR      NZ,ERROR2       ;NO - go process error
5212 211144     00300         LD      HL,4411H        ;Get Model 3 himem in HL
5215 223752     00310         LD      (GETMEM+2),HL   ;Patch initialization
5218 224652     00320         LD      (STRMEM+2),HL   ; for a III
521B 217B44     00330         LD      HL,447BH        ;Get M3 ENQUE in HL
521E 224152     00340         LD      (SETUP+1),HL    ; and patch init.
5221 3E14       00350         LD      A,14H           ;Get SP offset in A
5223 32AE52     00360         LD      (ADDER),A       ; and patch working part
5226 21D244     00370         LD      HL,44D2H        ;Get M3 SHOWBY in HL
5229 22B852     00380         LD      (WORD+1),HL     ; and patch working part
522C 21D744     00390         LD      HL,44D7H        ;Get M3 SHOWRD in HL
522F 22AB52     00400         LD      (BYTE+1),HL     ; and patch working part
                00410 ;
5232            00420 RELOC   EQU     $
                00430 ;
5232 21B952     00440         LD      HL,END          ;Get current END in HL
                00450 ;
5235            00460 GETMEM  EQU     $
                00470 ;
5235 ED5B4949   00480         LD      DE,(4949H)      ;Get HIMEM in DE
5239 011A00     00490         LD      BC,END-START+1  ;Get length of prog in BC
523C EDB8       00500         LDDR                    ;Move working part
523E D5         00510         PUSH    DE              ;Save new memory size
523F 13         00520         INC     DE              ;To point to START
                00530 ;
5240            00540 SETUP   EQU     $
                00550 ;
5240 CD1044     00560         CALL    4410H           ;CALL ENQUE to patch into
                00570                                 ;interrupt chain
5243 D1         00580         POP     DE              ;Get new HIMEM back
                00590 ;
5244            00600 STRMEM  EQU     $
                00610 ;
5244 ED534949   00620         LD      (4949H),DE      ; and put it into HIMEM
5248 C9         00630         RET                     ;RET to DOS READY
                00640 ;
5249            00650 ERROR1  EQU     $
                00660 ;
5249 215352     00670         LD      HL,MSG1         ;Point HL to MSG1
524C DD         00680         DEFB    0DDH            ;Make LD HL look like
                00690                                 ;     LD IX to save bytes
                00700 ;
```

```
5240            00710 ERROR2  EQU     $
                00720 ;
5240 217752     00730         LD      HL,MSG2         ;Point HL to MSG2
5243 C36447     00740         JP      4467H           ; and JP to display msg
                00750                                 ; and RET to DOS READY
                00760 ;
5253            00770 MSG1    EQU     $
                00780 ;
5253 54         00790         DEFM    'This program is for Newdos/80 only.'
     68 69 73 20 70 72 6F 67 72 61 6D 20 69 73 20 66
     6F 72 20 4E 65 77 64 6F 73 2F 38 30 20 6F 6E 6C
     79 2E
5276 0D         00800         DEFB    0DH
                00810 ;
5277            00820 MSG2    EQU     $
                00830 ;
5277 54         00840         DEFM    'This program is for Model I or III only.'
     68 69 73 20 70 72 6F 67 72 61 6D 20 69 73 20 66
     6F 72 20 4D 6F 64 65 6C 20 49 20 6F 72 20 49 49
     49 20 6F 6E 6C 79 2E
529F 0D         00850         DEFB    0DH
                00860 ;
52A0            00870 START   EQU     $
                00880 ;
                00890 ;Refer to section 3.8 on page 3-3 for information on the
                00900 ;following four bytes and their uses.
                00910 ;
52A0 0000       00920         DEFW    0000H           ;2 bytes for DOS's use
52A2 0101       00930         DEFW    0101H           ;2 bytes for counters
                00940 ;
52A4 3A0000     00950         LD      A,(FOLLOW)      ;Get byte to follow in A
52A7 212A3C     00960         LD      HL,3C2AH        ;Point to screen location
                00970 ;
52AA            00980 BYTE    EQU     $
                00990 ;
52AA CD6840     01000         CALL    4068H           ;CALL SHOWBY
                01010 ;
52AE            01020 ADDER   EQU     $+1
                01030 ;
52AD 21E000     01040         LD      HL,0EH          ;Get SP adder in HL
52B0 39         01050         ADD     HL,SP           ; add it to SP value
                01060                                 ; and store result in HL
52B1 5E         01070         LD      E,(HL)          ;Move HL
52B2 23         01080         INC     HL              ; over
52B3 56         01090         LD      D,(HL)          ;  to DE
52B4 212E3C     01100         LD      HL,3C2EH        ;Point HL to screen loc.
52B7            01110 WORD    EQU     $
52B7 C36340     01120         JP      4063H           ;JP to SHOWRD and RET
52B9            01130 END     EQU     $-1
5200            01140         END     INIT            ;Auto start at INIT
00000 TOTAL ERRORS
```

```
ADDER   52AE   BYTE    52AA   END     52B9   ERROR1  5249   ERROR2  5240
FOLLOW  0000   GETMEM  5235   INIT    5200   MSG1    5253   MSG2    5277
RELOC   5232   SETUP   5240   START   52A0   STRMEM  5244   WORD    52B7
```

FIXHIT/ASM:

Fix NEWDOS/80 version 2 (Model III) Hash Code Tables
by Tony Domigan

Following a repeated request from a friend to describe the method of reconstructing a NEWDOS/80 Hash Table, I decided to save myself time in the future by writing a program to do this relatively simple but tedious task - after all, this is where a micro excels. Super Utility plus does the job much better than my routine, however, SU+ does not support a double-sided NEWDOS/80 configuration.

FIXHIT is my first attempt at this task. It will reconstruct a HASH Code Table or HIT Sector, however, because of the limitation of my method, it may not be 100% correct.

LIMITATIONS

I chose to do File I/O, mainly because it was something I had not tried before. In order for this method to work the following conditions must be met:

1. The HIT Sector (relative sector 01 of DIR/SYS) and the DIR/SYS FPDE page (relative sector 03 of DIR/SYS) must be read-protected, and

2. The HASH code for DIR/SYS, i.e. C4, must be in relative byte 01 of the HIT sector.

Another problem occurs with FXDE's. I chose not to decode each FPDE for FXDE pointers. Therefore, I only check if the first byte of the FPDE = 90H (i.e. FXDE) and if so the old HASH code is moved to my new HASH table. If this code had been accidentally overwritten then the TABLE will have an error in it. I hope to fix this in my next version (if anyone requests the upgrade).

THE PROGRAM

Execute FIXHIT by the command:

FIXHIT [:][dn]

The colon is optional and if no drive number [dn] is specified then drive zero is assumed.

FIXHIT will read each sector of DIR/SYS and reconstruct a new Hash Code Table. As each sector is read a count is kept of any sectors not read-protected. When the new table is complete:

1. If no hash errors but there are read-protect errors, WRDIRP is executed.

2. If no hash or read-protect errors then exit to DOS.

3. If hash errors are found then you will be prompted for your intentions. If you answer Y (or y) to fix the disk then the new HIT sector will be written and WRDIRP is executed if any sectors were not read-protected. However, if you answer otherwise then you will abort to DOS immediately.

ENHANCEMENTS

FIXHIT satisfies my needs for the moment, but what I may do in the next version is:

1. Use SECTOR I/O in place of File I/O

2. Decode FPDE's and FXDE's properly and

3. Write my own WRDIRP loop (it would be much faster than WRDIRP).

Tony Domigan P.O. Box 150, Thomastown, Victoria, 3074, Australia.

```
                00100 ;             FIXHIT/ASM - Version 1.0
                00110 ;               NEWDOS80/V2  (III)
                00120 ;                 2nd February 1985
                00130 ;        For NORTHERN BYTES & the PUBLIC DOMAIN
                00140 ;                   by Tony Domigan
                00150 ;  PO Box 150, Thomastown, Victoria, 3074, Australia
                00160 ;MCI-ID:254-5121. SOURCE-ID:BCT039. TAB-ID:DOMIPOBONDN
                00170 ;-------------------------------------------------------
                00180 ;                 Pointers used in FIXHIT
                00190 ;  B  = FPDE counter      BC' = FPDE & Page counters
                00200 ;  DE = New HIT row ptr   DE' = New HIT column ptr
                00210 ;  HL = Buffer FPDE ptr   HL' = FCB buffer (first FPDE)
                00220 ;  IX = OLD HIT row ptr   IY  = OLD HIT column ptr
                00230 ;-------------------------------------------------------
                00240 ;
5200            00250         ORG     5200H           ;Anywhere abv 5200H
                00260 ;-------------------------------------------------------
                00270 ;       xxTEST FOR DRIVESPECxx
                00280 ;-------------------------------------------------------
5200            00290 PARSER  EQU     $
5200 7E         00300         LD      A,(HL)          ;Trailing chars?
5201 FE0D       00310         CP      0DH             ;Assume drive 0?
5203 281C       00320         JR      Z,BANNER        ;Yes, print msg
5205 FE3A       00330         CP      3AH             ;Colon
5207 2002       00340         JR      NZ,NUMBER       ;Skip if not colon
5209 23         00350         INC     HL              ;Yes, colon so ck next
520A 7E         00360         LD      A,(HL)          ;Get next character
520B FE34       00370 NUMBER  CP      34H             ;> Drive 3?
520D 3004       00380         JR      NC,BADNUM       ;Yes, bad drive number
520F FE30       00390         CP      30H             ;>=0and<=3
5211 3005       00400         JR      NC,POSTDR       ;Yes, use drive number
5213 3EA0       00410 BADNUM  LD      A,0A0H          ;Illegal drive number
5215 C30944     00420         JP      4409H           ;Error exit
5218 328854     00430 POSTDR  LD      (DRIVE),A       ;Post drv num to FCB
521B 32CF53     00440         LD      (DRVNUM),A      ;Post drv num to banner
521E 328854     00450         LD      (WRPDRV),A      ;Post drive to WRDIRP,n
                00460 ;-------------------------------------------------------
                00470 ;       xxDISPLAY BANNERxx
                00480 ;-------------------------------------------------------
5221            00490 BANNER  EQU     $
5221 CDC901     00500         CALL    01C9H           ;CLS
5224 21A453     00510         LD      HL,BANMSG       ;Pt to msg
5227 CD6744     00520         CALL    4467H           ;Display msg
                00530 ;-------------------------------------------------------
                00540 ;       xxOPEN DIR/SYS OF NOMINATED DRIVExx
                00550 ;-------------------------------------------------------
522A            00560 OPEN    EQU     $
```

```
522A 21D054   00570          LD    HL,BUFF1     ;FCB buffer
522D 11B054   00580          LD    DE,FCB       ;Pt to my FCB
5230 0600     00590          LD    B,00H        ;LRL=Full sector I/O
5232 CD2444   00600          CALL  4424H        ;OPEN   DIR/SYS
5235 C24C53   00610          JP    NZ,EREXIT    ;Z=No error
5238 3ABC54   00620          LD    A,(LEN)      ;File length from FCB
523B D602     00630          SUB   02H          ;REL Count=EOF-HIT
523D 4F       00640          LD    C,A          ;Post page counter to C
              00650 ;--------------------------------------------------
              00660 ;        xxSETUP POINTERSxx
              00670 ;--------------------------------------------------
523E          00680 SETUP    EQU   $
523E 0608     00690          LD    B,08H        ;FPDE Slots 0-7
5240 11D055   00700          LD    DE,BUFF2     ;New HIT Buffer
5243 DD21D056 00710          LD    IX,BUFF3     ;Old HIT Buffer
5247 C5       00720          PUSH  BC           ;Save slot/page count
5248 D5       00730          PUSH  DE           ;Save new HIT buff addr
5249 E5       00740          PUSH  HL           ;Save FCB buffer addr
524A DDE5     00750          PUSH  IX           ;Save old HIT buff addr
524C FDE1     00760          POP   IY           ;IY=IX
524E D9       00770          EXX                ;Switch to AF',BC',HL'
524F E1       00780          POP   HL           ;HL' = HL
5250 D1       00790          POP   DE           ;DE' = DE
5251 C1       00800          POP   BC           ;BC' = BC
5252 D9       00810          EXX                ;Switch to orig set
              00820 ;--------------------------------------------------
              00830 ;        xxPOSITION TO HIT SECTORxx
              00840 ;--------------------------------------------------
5253          00850 INIT     EQU   $
5253 CD6E53   00860          CALL  READ         ;Read GAT sector
5256 CD6E53   00870          CALL  READ         ;Read HIT sector
              00880 ;--------------------------------------------------
              00890 ;        xxPREPARE BUFFERSxx
              00900 ;--------------------------------------------------
5259          00910 SETBUF   EQU   $
5259 C5       00920          PUSH  BC           ;Save counters
525A E5       00930          PUSH  HL           ;Save FCB buff addr
525B D5       00940          PUSH  DE           ;Save BUFF1 addr
525C D5       00950          PUSH  DE           ;Again
525D DDE5     00960          PUSH  IX           ;Move IX ..
525F D1       00970          POP   DE           ;to DE (buff3)
5260 010001   00980          LD    BC,256       ;Full sector
5263 C5       00990          PUSH  BC           ;Save it
5264 EDB0     01000 MOVOLD   LDIR               ;Mv orig buffer to buff3
5266 C1       01010          POP   BC           ;Ret byte count
5267 08       01020          DEC   BC           ;Minus 1
5268 D1       01030          POP   DE           ;Save buff2 addr
5269 13       01040          INC   DE           ;Plus 1
526A 3600     01050          LD    (HL),00H     ;Load 0 to 1st byte
526C EDB0     01060 MAKNEW   LDIR               ;Zero buffer 2
526E D1       01070          POP   DE           ;Restore buff3 addr
526F E1       01080          POP   HL           ;Restore FCB buff1 addr
5270 C1       01090          POP   BC           ;Restore FPDE/PAGE count
5271 3ABC54   01100          LD    A,(LEN)      ;Length of DIR/SYS
5274 D60A     01110          SUB   10           ;= Extra allocation
5276 32EF55   01120          LD    (BUFF2+1FH),A ;Replace allocation
              01130 ;--------------------------------------------------
              01140 ;        xxBEGIN LOOP FPDE PAGESxx
              01150 ;--------------------------------------------------
5279          01160 PAGE     EQU   $
5279 CD6E53   01170          CALL  READ         ;Read fpde page to buff1
              01180 ;--------------------------------------------------
              01190 ;        xxLOOP FOR 8 FPDE SLOTS PER FPDE PAGE xx
              01200 ;--------------------------------------------------
527C          01210 FPDE     EQU   $
527C 7E       01220          LD    A,(HL)       ;1st byte of fpde
527D FE00     01230          CP    00H          ;Not allocated?
527F 281F     01240          JR    Z,NXSLOT     ;Skip if not allocated
5281 FE90     01250          CP    90H          ;FXDE?
5283 2005     01260          JR    NZ,HASHIT    ;No, make hash code
5285 DD7E00   01270          LD    A,(IX+0)     ;Use orig hash code
5288 1815     01280          JR    FIXIT        ;Post to new HIT
              01290 ;--------------------------------------------------
              01300 ;        xxMAKE HASH CODE FROM FILESPECxx
              01310 ;--------------------------------------------------
528A          01320 HASHIT   EQU   $
528A E5       01330          PUSH  HL           ;Save slot pointer
528B C5       01340          PUSH  BC           ;Save slot counter
528C 7D       01350          LD    A,L          ;LSB of FPDE
528D C605     01360          ADD   A,05H        ;Increment by five
528F 6F       01370          LD    L,A          ;HL-->Filespec
5290 0608     01380 STHASH   LD    B,08H        ;Filsespec & Ext
5292 AF       01390          XOR   A            ;Zero accum
5293 AE       01400 HLOOP    XOR   (HL)         ;XOR (HL) with accum
5294 23       01410          INC   HL           ;Next char in filespec
5295 07       01420          RLCA               ;Rotate left result in A
5296 10FB     01430          DJNZ  HLOOP        ;Loop for 11 characters
5298 FE00     01440          CP    00H          ;Hash = 0
529A 2001     01450          JR    NZ,HASHOK    ;Skip if not 0
529C 3C       01460          INC   A            ;Zero not allowed
529D C1       01470 HASHOK   POP   BC           ;Restore slot counter
529E E1       01480          POP   HL           ;Restore FPDE pointer
529F 12       01490 FIXIT    LD    (DE),A       ;Post code to new HIT
              01500 ;--------------------------------------------------
              01510 ;        xxPOSITION TO NEXT FPDE SLOT IN PAGExx
              01520 ;--------------------------------------------------
52A0          01530 NXSLOT   EQU   $
52A0 C5       01540          PUSH  BC           ;Save slot counter
52A1 D5       01550          PUSH  DE           ;Move new HIT row counter
52A2 C1       01560          POP   BC           ;To BC
52A3 112000   01570          LD    DE,20H       ;Incr all slots by 32
52A6 19       01580          ADD   HL,DE        ;FCB row = +32
52A7 E5       01590          PUSH  HL           ;Save HL
52A8 DD19     01600          ADD   IX,DE        ;Old HIT row = +32
52AA C5       01610          PUSH  BC           ;Move old DE to
52AB E1       01620          POP   HL           ;to HL so that
52AC 19       01630          ADD   HL,DE        ;New HIT = +32
52AD EB       01640          EX    DE,HL        ;Xfer back to DE
52AE E1       01650          POP   HL           ;Recover FCB slot addr
52AF C1       01660          POP   BC           ;Recover slot count
52B0 10CA     01670          DJNZ  FPDE         ;Loop for 8 slots
              01680 ;--------------------------------------------------
              01690 ;        xxLOOP FOR ALL FPDE PAGESxx
              01700 ;--------------------------------------------------
52B2          01710 NXFPDE   EQU   $
52B2 D9       01720          EXX                ;AF', BC', HL'
52B3 0D       01730          DEC   C            ;Decrement page count
52B4 2810     01740          JR    Z,HITCK      ;Finished, check hit sr's
52B6 13       01750          INC   DE           ;Bump new hit column ptr
52B7 FD23     01760          INC   IY           ;Bump old hit column ptr
52B9 C5       01770          PUSH  BC           ;Save
52BA D5       01780          PUSH  DE           ; new
52BB E5       01790          PUSH  HL           ;  pointers
52BC FDE5     01800          PUSH  IY           ;   to stack
52BE D9       01810          EXX                ;Non-prime registers
52BF DDE1     01820          POP   IX           ;Restore
52C1 E1       01830          POP   HL           ; New pointers
52C2 D1       01840          POP   DE           ;  To the native
52C3 C1       01850          POP   BC           ;   Registers
52C4 18B3     01860          JR    PAGE         ;Loop till next page
              01870 ;--------------------------------------------------
              01880 ;        xxCOMPARE OLD & NEW HIT SECTORSxx
              01890 ;--------------------------------------------------
52C6          01900 HITCK    EQU   $
52C6 DD210000 01910          LD    IX,0000H     ;Non-match counter
52CA 21D055   01920          LD    HL,BUFF2     ;New hash table
52CD 11D056   01930          LD    DE,BUFF3     ;Old hash table
52D0 0600     01940          LD    B,00H        ;256 bytes
52D2 1A       01950 CKLOOP   LD    A,(DE)       ;Get old hash code
52D3 BE       01960          CP    (HL)         ;Compare with new
52D4 2802     01970          JR    Z,SAME       ;Skip if the same
52D6 DD23     01980          INC   IX           ;Bump error count
52D8 23       01990 SAME     INC   HL           ;Next new hash code
52D9 13       02000          INC   DE           ;Next old hash code
52DA 10F6     02010          DJNZ  CKLOOP       ;Loop for whole table
52DC DDE5     02020          PUSH  IX           ;Move error count
52DE D1       02030          POP   DE           ;to DE
52DF 7B       02040          LD    A,E          ;Errors always <256
52E0 21D253   02050          LD    HL,ERRNUM    ;Point to error num $
52E3 F5       02060          PUSH  AF           ;Save error count
52E4 CD0744   02070          CALL  4407H        ;Load to string
52E7 21D253   02080          LD    HL,ERRNUM    ;Point to error msg
52EA CD6744   02090          CALL  4467H        ;Display num errors
52ED F1       02100          POP   AF           ;Restore num errors
52EE FE00     02110          CP    00H          ;No errors?
52F0 285F     02120          JR    Z,CLOSE      ;Yes, omit fix rtne
52F2 21E953   02130          LD    HL,WRPRMT    ;Fix Y/N Msg
52F5 CD6744   02140          CALL  4467H        ;Display msg
52F8 CD4900   02150          CALL  49H          ;KBWAIT
52FB F620     02160          OR    20H          ;Cvrt to l/c
52FD FE79     02170          CP    79H          ;yes?
52FF 280C     02180          JR    Z,YES        ;then fixit
```

```
5301 AF          02190          XOR     A               ;Zero accum
5302 329A54      02200          LD      (RCOUNT),A      ;Stop RP fix
5305 21A054      02210 NO       LD      HL,NOMSG        ;Abort msg
5308 CD6744      02220          CALL    4467H           ;Display it
530B 1844        02230          JR      CLOSE           ;JP over write rtne
530D 21B854      02240 YES      LD      HL,WRMSG        ;Write message
5310 CD6744      02250          CALL    4467H           ;Display it
                 02260 ;-------------------------------------------------
                 02270 ;        xx POSITION TO HIT SECTORxx
                 02280 ;-------------------------------------------------
5313 11B054      02290          LD      DE,FCB          ;Pt to FCB
5316 21D054      02300          LD      HL,BUFF1        ;Pt to FCB buffer
5319 3EC9        02310          LD      A,0C9H          ;Patch RET NZ to RET
531B 32BC53      02320          LD      (NOERR),A       ;To stop dupe err msg
531E CD3F44      02330 SOF      CALL    443FH           ;Move to start of file
5321 CD6E53      02340 GAT      CALL    READ            ;Read GAT
                 02350 ;-------------------------------------------------
                 02360 ;        xxMOVE NEW HIT TO FCB BUFFERxx
                 02370 ;-------------------------------------------------
5324 21D055      02380 MOVE     LD      HL,BUFF2        ;New HIT buffer
5327 11D054      02390          LD      DE,BUFF1        ;FCB buffer
532A 010001      02400          LD      BC,256          ;Bytes to move
532D EDB0        02410          LDIR                    ;Xfer it
                 02420 ;-------------------------------------------------
                 02430 ;        xxPREPARE FCB FOR READ-PROTECT WRITExx
                 02440 ;-------------------------------------------------
532F 3AB054      02450          LD      A,(FCB)         ;FCB 1st byte
5332 F601        02460          OR      01H             ;Make read-protect
5334 32B054      02470          LD      (FCB),A         ;Ammend FCB
5337 3AB154      02480          LD      A,(FCB+1)       ;FCB 2nd byte
533A F640        02490          OR      40H             ;Do not update EOF
533C 32B154      02500          LD      (FCB+1),A       ;Ammend FCB
                 02510 ;-------------------------------------------------
                 02520 ;        xxWRITE WITH VERIFYxx
                 02530 ;-------------------------------------------------
533F             02540 WRITE    EQU     $
533F 21D054      02550          LD      HL,BUFF1        ;FCB buffer
5342 11B054      02560          LD      DE,FCB          ;DE --> FCB
5345 0600        02570          LD      B,00H           ;LRL=256
5347 CD3C44      02580          CALL    443CH           ;Write with verify
534A 2805        02590          JR      Z,CLOSE         ;Exit if no error
534C F680        02600 EREXIT   OR      80H             ;Make long err msg
534E CD0944      02610          CALL    4409H           ;Display error msg
                 02620 ;-------------------------------------------------
                 02630 ;        xxCLOSE FCBxx
                 02640 ;-------------------------------------------------
5351             02650 CLOSE    EQU     $
5351 CD2844      02660          CALL    4428H           ;Close FCB
5354 3A9A54      02670          LD      A,(RCOUNT)      ;How many rp errs?
5357 FE00        02680          CP      00H             ;No errors?
5359 CA2040      02690          JP      Z,4020H         ;Exit if so
535C 219A54      02700          LD      HL,RCOUNT       ;Rp err no $
535F CD0744      02710          CALL    4407H           ;Move to $
5362 21A054      02720          LD      HL,RPTMSG       ;Pt to err num $
5365 CD6744      02730          CALL    4467H           ;Disp msg
5368 218154      02740          LD      HL,WRDIRP       ;WRDIRP CMD
536B C32D44      02750          JP      442DH           ;Execute WRDIRP,n
                 02760 ;-------------------------------------------------
                 02770 ;        xxREAD SECTORxx
                 02780 ;-------------------------------------------------
536E             02790 READ     EQU     $
536E C5          02800          PUSH    BC              ;Save slot count
536F E5          02810          PUSH    HL              ;Save slot pointer
5370 D5          02820          PUSH    DE              ;Save new hit ptr
5371 21D054      02830          LD      HL,BUFF1        ;FCB buffer
5374 11B054      02840          LD      DE,FCB          ;FCB
5377 010000      02850          LD      BC,0000H        ;LRL=256
537A CD3644      02860          CALL    4436H           ;Read sector
537D D1          02870          POP     DE              ;Restore new row ptr
537E E1          02880          POP     HL              ;Restore fcb row ptr
537F C1          02890          POP     BC              ;Restore slot count
5380 F5          02900          PUSH    AF              ;Save error count
5381 FE1C        02910          CP      1CH             ;EOF?
5383 CA4C53      02920          JP      Z,EREXIT        ;Error exit
5386 FE1D        02930          CP      1DH             ;Past EOF?
5388 CA4C53      02940          JP      Z,EREXIT        ;Error exit
538B F1          02950          POP     AF              ;Restore error code
538C C0          02960 NOERR    RET     NZ              ;Continue if NZ
                 02970 ;-------------------------------------------------
                 02980 ;        xxREAD-PROTECT ERRORxx
                 02990 ;-------------------------------------------------
538D             03000 WRPERR   EQU     $
538D 3ABA54      03010          LD      A,(NEXT)        ;Get next pointer
5390 3D          03020          DEC     A               ;Pt to last sector read
5391 E5          03030          PUSH    HL              ;Save FCB ptr
5392 216754      03040          LD      HL,RPSR         ;Sector num
5395 CD0744      03050          CALL    4407H           ;Post to $
5398 215954      03060          LD      HL,NRPMSG       ;Nr msg
539B CD6744      03070          CALL    4467H           ;Display it
539E 219A54      03080          LD      HL,RCOUNT       ;NR error count
53A1 34          03090          INC     (HL)            ;Bump counter
53A2 E1          03100          POP     HL              ;Restore FCB ptr
53A3 C9          03110          RET                     ;Continue processing
                 03120 ;-------------------------------------------------
                 03130 ;        xxSTRINGS AND STORAGExx
                 03140 ;-------------------------------------------------
53A4 46          03150 BANMSG   DEFM    'FIXHIT for NEWDOS80/V2 - Drive on Test --> '
     49 58 48 49 54 20 66 6F 72 20 4E 45 57 44 4F 53
     38 30 2F 56 32 20 2D 20 44 72 69 76 65 20 6F 6E
     20 54 65 73 74 20 2D 2D 3E 20
53CF 30          03160 DRVNUM   DEFM    '0'
53D0 0A          03170          DEFB    0AH
53D1 0D          03180          DEFB    0DH
53D2 20          03190 ERRNUM   DEFM    '  H '
     20 48 20
53D6 48          03200          DEFM    'Hash code Errors '
     61 73 68 20 63 6F 64 65 20 45 72 72 6F 72 73 20
53E7 0A          03210          DEFB    0AH
53E8 0D          03220          DEFB    0DH
53E9 52          03230 WRPRMT   DEFM    'Reply Y/N to Repair Diskette Hash Code Table -->'
     65 70 6C 79 20 59 2F 4E 20 74 6F 20 52 65 70 61
     69 72 20 44 69 73 6B 65 74 74 65 20 48 61 73 68
     20 43 6F 64 65 20 54 61 62 6C 65 20 2D 20 3E
5419 0A          03240          DEFB    0AH
541A 0D          03250          DEFB    0DH
541B 57          03260 WRMSG    DEFM    'Writing the Corrected Hit Sector Now'
     72 69 74 69 6E 67 20 74 68 65 20 43 6F 72 72 65
     63 74 65 64 20 48 69 74 20 53 65 63 74 6F 72 20
     4E 6F 77
543F 0D          03270          DEFB    0DH
5440 2A          03280 NOMSG    DEFM    'xxABORTINGxx Per Request'
     2A 41 42 4F 52 54 49 4E 47 2A 2A 20 50 65 72 20
     52 65 71 75 65 73 74
5458 0D          03290          DEFB    0DH
5459 44          03300 NRPMSG   DEFM    'DIR/SYS FRS : '
     49 52 2F 53 59 53 20 46 52 53 20 3A 20
5467 30          03310 RPSR     DEFM    '00 '
     30 20
546A 69          03320          DEFM    'is NOT READ-PROTECTED'
     73 20 4E 4F 54 20 52 45 41 44 2D 50 52 4F 54 45
     43 54 45 44
547F 0A          03330          DEFB    0AH
5480 0D          03340          DEFB    0DH
5481 57          03350 WRDIRP   DEFM    'WRDIRP,'
     52 44 49 52 50 2C
5488 30          03360 WRPDRV   DEFB    30H
5489 0D          03370          DEFB    0DH
548A 52          03380 RPTMSG   DEFM    'Read-Protecting '
     65 61 64 2D 50 72 6F 74 65 63 74 69 6E 67 20
549A 00          03390 RCOUNT   DEFB    00H
549B 00          03400          DEFB    00H
549C 48          03410          DEFM    'H '
     20
549E 44          03420          DEFM    'Directory Sectors'
     69 72 65 63 74 6F 72 79 20 53 65 63 74 6F 72 73
54AF 0D          03430          DEFB    0DH
                 03440 ;-------------------------------------------------
                 03450 ;        xx F C B xx
                 03460 ;-------------------------------------------------
54B0 44          03470 FCB      DEFM    'DIR/SYS:'      ;Filespec
     49 52 2F 53 59 53 3A
54B8 30          03480 DRIVE    DEFM    '0'             ;Drivespec
54B9 0D          03490          DEFB    0DH
54BA 00          03500 NEXT     DEFB    00H             ;Next sector pointer
54BB 00          03510          DEFB    00H
54BC 00          03520 LEN      DEFB    00H             ;LSB length of file
54BD 00          03530          DEFB    00H
54BE 00          03540 LUMP     DEFB    00H             ;Starting lump
54BF 00          03550 GRAN     DEFB    00H             ;Number of grans
0010             03560 EXT      DEFS    10H             ;Extents
                 03570 ;-------------------------------------------------
                 03580 ;        xxSTORAGE BUFFERSxx
```

```
                03570 ;-------------------------------------------------
0100            03600 BUFF1   DEFS    256             ;FCB buffer
0100            03610 BUFF2   DEFS    256             ;New HASH CODE buffer
0100            03620 BUFF3   DEFS    256             ;Orig HIT sector
5200            03630         END     PARSER
00000 TOTAL ERRORS

BADNUM  5213    BANMSG  5304    BANNER  5221    BUFF1   5400    BUFF2   5500
BUFF3   5600    CKLOOP  5202    CLOSE   5351    DRIVE   5488    DRVNUM  53CF
ERRXIT  534C    ERRNUM  5302    E.T     54C0    FCB     5480    FIXIT   529F
FPDE    527C    GAT     5321    GRAN    549F    HASHIT  528A    HASHOK  529D
HITCK   52C6    HLOOP   5293    INIT    5253    LEN     54BC    LUMP    54BE
MAKNEW  526C    MOVE    5324    MOVOLD  5264    NEXT    54BA    NO      5305
NOERR   539C    NRPERR  5390    NRPMSG  5459    NUMSER  5208    NWMSG   5440
NXFPDE  52B2    NXSLOT  52A0    OPEN    522A    PAGE    5279    PARSER  5200
POSTDR  5218    RCOUNT  549A    READ    536E    RPSR    5467    RPTMSG  548A
SAME    52D0    SETBUF  5259    SETUP   523E    SOF     531E    STHASH  5290
WRDIRP  5481    WRITE   533F    WRMSG   541B    WRPDRV  5498    WRPRMT  53E9
YES     5300
```

ANCHOR SIGNALMAN MARK XII DTR FIX

[EDITOR'S NOTE: This information has been provided by Guy Omer, SYSOP of the 8/N/1 #001 BBS (the BBS phone number is (904) 377-1200). Unfortunately, the instructions below are not as clear as I might wish, but I believe that the subject matter will be of great interest to many owners of Mark XII Modems. If any NORTHERN BYTES reader would like to write instructions that are a bit clearer, or better yet, draw a pictorial diagram showing this modification, we would like to publish it in these pages. In the meantime, please consider the instructions below as incomplete, and for use by EXPERIENCED HARDWARE HACKERS ONLY!!! NORTHERN BYTES assumes NO LIABILITY WHATSOEVER should you attempt to use these instructions and damage your Modem or any other part of your system!!!!!]

The Anchor Signalman Mark XII does not support the RS-232 DTR (data terminal ready) signal. Too bad, since at $250 (discounted) this is probably the cheapest stand-alone 1200 bps modem available, and seems to work just fine otherwise. Unfortunately, quite a bit of software (BBSs especially) depend on DTR to hang up the phone.

This file describes a hardware fix which will add DTR to the Mark XII. Obviously, this voids your 2-year warranty. However, the same thing COULD be done with an external box.

You will need:

- 1 - 2N4401 transistor
- 1 - 10K ohm 1/4 watt resistor
- a short piece of thin wire (#30 wirewrap wire suggested)
- shrink-wrap & tape (to do it right)
- an exacto knife
- solder, soldering iron, and the ability to use it

1) Using a flat-blade screwdriver, open the Signalman case. Try not to mangle it too badly.

2) Remove the circuit board and turn it so that the solder side is facing up, and the serial cable is at the bottom.

3) It should say "Anchor Automation Inc 00472 Rev A" in the upper right-hand corner. If it doesn't, this may not work, or may not make sense.

4) In the lower right-hand corner of the board, just to the left and above the right-hand phone connector, you will notice two vertical groups of three pads. The left group has no traces coming from it. The right group has traces coming from the bottom and top pads.

5) Cut the trace coming from the bottom right-hand pad. Try to do this near to the pad.

6) Position the 2N4401 transistor with the flat side toward the board, and the wires toward the top of the board.

7) Solder the RIGHT-hand wire of the transistor to the lower right-hand pad of the group of six that you located above. (The one that you cut the trace to.)

8) Solder the MIDDLE wire of the transistor to the cut trace, on the other side of the cut.

9) Solder one end of the 10K resistor to the remaining wire of the transistor.

10) Solder a 3" piece of wire to the other end of the 10K resistor.

11) Slip a piece of heat-shrink over the resistor, and shrink.

12) Now, find the group of pads coming from the RS-232 cable. Find the 4th pad from the left, and 2nd from the bottom. There should be traces coming from the pads just above and below it.

13) Solder the other end of the wire to this pad.

14) Put a piece of tape over the whole shebang, just to keep it from moving.

15) That's it! The modem should now observe DTR. If the DTR signal is 0 or negative voltage, the telephone line (through the on-board relay) will be disconnected from the modem. If the DTR signal is above a couple of volts then everything works normally.

16) Make sure that you connect DTR to your computer and that your computer asserts DTR! The modem will NOT operate without DTR asserted!

17) If DTR is dropped, the line to the modem will be dropped. The modem should notice it and drop carrier detect to your computer, as well as decide to hang up on its very own. At this point, you can bring DTR back up, and the caller (or callee) should be gone. I have no idea what the timing should be - whatever it takes the relay to operate. 1/10 second should be more than ample.

18) You will soon discover why people with Hayes modems are constantly pulling the front off to operate the DIP switches. PC-Talk (and maybe other programs) insists on dropping DTR when you exit or change communication parameters. Oh well, you can't have everything!

19) Oh yeah - quit gawking and put the thing back together.

20) Pray. [Editor's note: I suggest you do this before you begin this project!!!]

INSTALLING TURBO PASCAL ON THE TRS-80 MODEL 4

by John T. Phillipp

There has been a lot of interest in structured programming lately, especially Pascal, but it has been hard to find a good Pascal compiler that will run on the TRS-80.

Borland International's TURBO PASCAL is an excellent buy. It has all the features of standard Pascal, plus a number of extensions, and retails for only $49.95. It compiles to Z-80 machine code, not P-code, so programs run very fast. It will run on the Model 4 under CP/M.

TURBO comes with a "Terminal Installation Program", and must be installed on your computer before it can be used. I had a devil of a time with that, since the TRS-80 Model 4 is NOT on their list, and the folks at the Borland technical support seem to think "computer" is spelled "I-B-M". I finally got it figured out with a lot of help from Mike Winterer, one of my fellow members at SAGATUG (the San Gabriel Valley TRS-80 User's Group).

Here are the directions to get TURBO Pascal installed on a TRS-80 Model 4 running Montezuma Micro CP/M 2.2 version 1.4:

1) Enter the TINST program: TINST <ENTER>
2) Choose option <S> (screen installation)
3) Enter #4 (ADM-3 terminal) to "Which Terminal"
4) Enter "YES" to "Do you want to modify the definition before installation"

The terminal specifications will be displayed one by one. Most will be left as-is, so just press <ENTER>

5) Terminal type: Change to "TRS-80 Model 4 CP/M"
6) Send a reset string to the terminal: Change to "Y"
7) Reset string: Change to 14 26
8) ERASE TO END OF LINE: Change to 21
9) START HIGHLIGHTING command: Change to 22
10) END HIGHLIGHTING command: Change to 22

The rest of the installation procedure follows the TURBO reference manual. With these codes, the LowVideo procedure will put the screen in reverse video until it is cancelled by the NormVideo procedure. The 22 code acts a toggle (thanks, Mike).

I hope that this of use to the readers of Northern Bytes!

NEWDOS/80 BUG

Reported by Greg Small

While creating a message to send to Tony Domigan in Australia via MCI Mail, Greg found a bug in NEWDOS/80 that he says he has never seen documented. As an example of this bug, name a file TONY/TXT. Then try to RENAME TONY/TXT TONY1/TXT. It appears that it is impossible to rename any file to another filename that starts with the letters "TO" without explicitly using the separator TO, because NEWDOS/80 will pick up the first two letters of the filename and treat them as the separator TO. This happens on both the Models I and III. It is possible to use the RENAME command properly provided the separator is included (in other words, RENAME TONY/TXT TO TONY1/TXT will work). Using a comma in place of the word "TO" won't work either.

Model 4 owners:

Big Blue ain't got NOTHIN' on YOU!

You DON'T need an IBM to run IBM software! There are dozens of programs that will convert TRS-80 software for execution on the IBM PC and compatibles. But the TRS-80 Model 4 is one of the most versatile machines available. There is a ton of software available for the Model III and the CP/M modes of the Model 4. There are more packages appearing every day for the Model 4 mode. And now, you can run a substantial portion of software written for the IBM on your lowly TRS-80 Model 4!

How? With Gee-Whiz Convert, a new utility from Dennis Allen and The Alternate Source. GW-Convert is a collection of programs and subroutines designed to translate IBM BASIC programs for execution on the TRS-80 Model 4 (in the 4 mode only). Plus, if you have either the Radio Shack or Grafyx Solution high resolution board, EVEN GRAPHICS ARE TRANSLATED. A high-res board is NOT required to use GW-Convert, but it is required if you want the graphics coverted. Persons using the Radio Shack high-resolution board may require the Grafyx Solution software for proper operation. This is available from The Alternate Source.

Some of the many IBM BASIC commands are converted include BEEP, CIRCLE, COLOR, CSRLIN, DATE$, DRAW, FILES, GET, INPUT, LINE, LOCATE, OPEN, PAINT, POINT, PRESET, PUT, RANDOMIZE, RESET, RND, SCREEN, TIME$ and WIDTH. Programming hints and tips are included for special handling of the ASCII character set, CIRCLE, CKLS, communications, DRAW, graphics, INKEY$, INPUT, joystick commands, KEY function, SCREEN function, softkeys and the VIEW command. GW-Convert includes several functions and subroutines that may be merged with your new Model 4 programs and sold without any royalty. GW-Convert is a must for software houses and programmers that want their software to be compatible with as many machines as possible.

GW-Convert includes a 50-page, indexed manual and all software needed to expand your TRS-80 program library with the thousands of IBM programs in the public domain and otherwise. The entire package is only $99.95 and available immediately from The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan, 48906. Telephone information is available by phoning (517) 482-8270. Although the documentation contains much technical programmer information, it is assumed that the user has some knowledge of BASIC. A knowledge of high-resolution commands may also be needed. Programs that use the IBM communication protocol may also require a BASIC compiler to execute properly on the Model 4.

The Grafyx Solution High Resolution Board for the Model 4 is $199; Grafyx Solution software, GBASIC 3.0 (for use with a Radio Shack high-res board) is $49.95. Software is included with the Grafyx Solution h/r board. Both are available from TAS.

The EDM Macro COMPiler

Many of you are familiar with the slick full-screen editing concepts available with EDX, ALE and EDM. Particularly with EDM and ALE, it may be necessary to write small "macro" programs to handle your specialized editing tasks. The EDM macro language is normally "low level", that is, the commands may look like 0V0R or "ZB0010. This is no longer necessary thanks to the EDM Macro COMPiler.

The EDM Macro COMPiler is now available. With this unique tool, you can write programs using a high level language. For example, high level commands may look like "CLEAR CRT" and "KILL BUFFER". The EDM Macro language offers many powerful commands that control program flow, editing, arithmetic operations, INPUT/OUTPUT, Screen Control, Storage, String Manipulation and other necessary items.

The EDM Macro COMPiler contains a diskette FULL of sample macro programs that have a wide variety of uses. All samples are carefully documented in a 100+ page manual. The index details which sample macros use each command and the corresponding page number where a listing of that macro can be found. Writing editing programs will never be easier.

The EDM Macro COMPiler, with documentation, is only $49.95 and available immediately from The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan, 48906 or phone (517) 482-8270. Please note that EDM (EDitor with Macros) is required to utilize this package and is available separately for $99.95. EDM is recommended for anyone who does serious editing. EDM is essential for anyone who deals with a variety of "filetypes". ALE and EDX purchasers can apply the entire cost of either of these packages toward the purchase of EDM.

NORTHERN BYTES
c/o Jack Decker
1804 West 18th Street
Lot # 155
Sault Ste. Marie, Michigan 49783
MCI Mail Address: 102-7413
Telex: 6501027413
(Answerback: 6501027413 MCI)

POSTMASTER: If undeliverable return to:
The Alternate Source, 704 N. Pennsylvania, Lansing, MI 48906

To: